Python Challenge SwissBlock

Jonathan Pham

October 16, 2022

1 Introduction

Please find attached the Python challenge files.

1) There is directly the notebook where you just need to run all the cells.

2) I also made the Python file, so you can run and get the result directly from the Terminal/Bash shell. For this second option, you need to install the packages with pip first. In this case, please do: "pip install -r requirements.txt" before running the python file.

I spent an afternoon on the exercice plus writing this pdf (around 4/5 hours).

2 My Guide Line / ideas:

The exercise is classic. First, I needed to import the data (OHLCV) directly from a crypto-exchange.

I chose Binance, as it is the biggest exchange), to get more relevant data (you can switch to FTX, kucoin etc).

I created input variables so that the user can directly define the date range, RSI parameters, time interval etc. After getting my OHLCV dataframe, I can truly start. For all trading strategies that I built, I am doing the same "task order guide line".

At this point, it's time to define my technical indicators. In our case, I just took the RSI (and the MFI which correlates the RSI and volume). note: I used directly the "technical analyses" library (ta) to get the cleanest and shortest code possible to optimize. We could also calculate the RSI in the same way by making a loop for each date, using the RSI formula and append this in our dataframe.

I'm printing the dataframe in a .csv file; this is useful for the python file, but for the notebook I recommend printing it directly.

Then I would like to present the RSI. Getting the RSI value in a data frame doesn't mean anything. So I decided to plot the RSI with respect to time and add the "over-buy/sell threshold". The graph allows us to interpret when the RSI is overbought/sold; divergence, momentum trend, etc.

In parallel I draw a "histogram" for the volume where it gives the volume with respect to time and I add in red line its average during this range of time.

Combining volume and RSI, we can see for example the moments when the market was volatile with a high volume traded and see what the dynamics of the RSI was at that time.
Then I simply created a loop to calculate how often the RSI was below or above the threshold, as you requested. Finally, I defined a "little strategy" combining RSI and volume. The idea was to trade when volume is high (average(volume) + σ(volume)) and the RSI is in the extreme zone, to bet on a potential market rebound.

3 Difficulties:

I had no specific trouble with this exercise, except for the input command at the beginning. I would make something "clean" so that if the user didn't write correctly, he should correct it, and not continue the code for getting compilation error later. Finally, I found a solution by using "datetime" as shown in my code.

4 Limitis and further ideas:

We can still improve a lot of things. For example, getting a volume histogram for each day of the week and adding the DATV (daily average traded volume) on it. It can permit to know which day (or times) of the week, a certain coin, seems to be the most volatile to take advantage of the liquidity and trend movement. From a technical point of view, we should implement a way that in my data frame it can recognize the day corresponding to the date (even if it is not in daily timeframe). I'm sure there is a way to do this.
The best part would be to define a trading strategy and backtest it; get an output file with : P/L, drawdown, sharp ratio, average winning/losing trades, win rate, etc. Since I have my dataframe, I really like to develop new ideas, analyses based on this. What's more, we can "code" in a cleaner way by getting a main file and define strategy, backtesting function into class on others files for instance.

Another thing, which I usually do is to define a json file where I can make a list of different coins, and then from the main file, I just need to call this file with the coin I want. This can also allow you to build a strategy based on multi-coins.
Finally, there is a whole optimization part that is needed. How do we know if the parameter we have chosen for the RSI is the best one for this coin? So we can also run an algorithm that, given a certain strategy, can set different parameters over a certain range with a certain step and list for which set of parameters the strategy was most effective (and taking care to not over-optimize !).