LygoBot

Riccardo Girgenti, Francesco Ligorio

September 2022 - June 2024

Introduction and Objective

The objective of the LygoBot project was to automate cryptocurrency trades on Binance, a well-known crypto exchange platform. We chose to code in Python due to the availability of robust third-party API packages. The bot's decisions on whether to buy or sell are based on a strategy we developed, which evolved throughout the project's completion.

Note that all strategies are based on the assumption that there is a correlation between the future price of an asset and its current price.

First Strategy

The first strategy was based on the RSI (Relative Strength Index) indicator, which provides useful indications of the relative strength of market momentum. The bot continuously calculated the RSI value on a one-minute timeframe, entering a position when the RSI was low and exiting after reaching a specific target price. This target could either result in a positive (take profit) or negative (stop loss) cash flow.

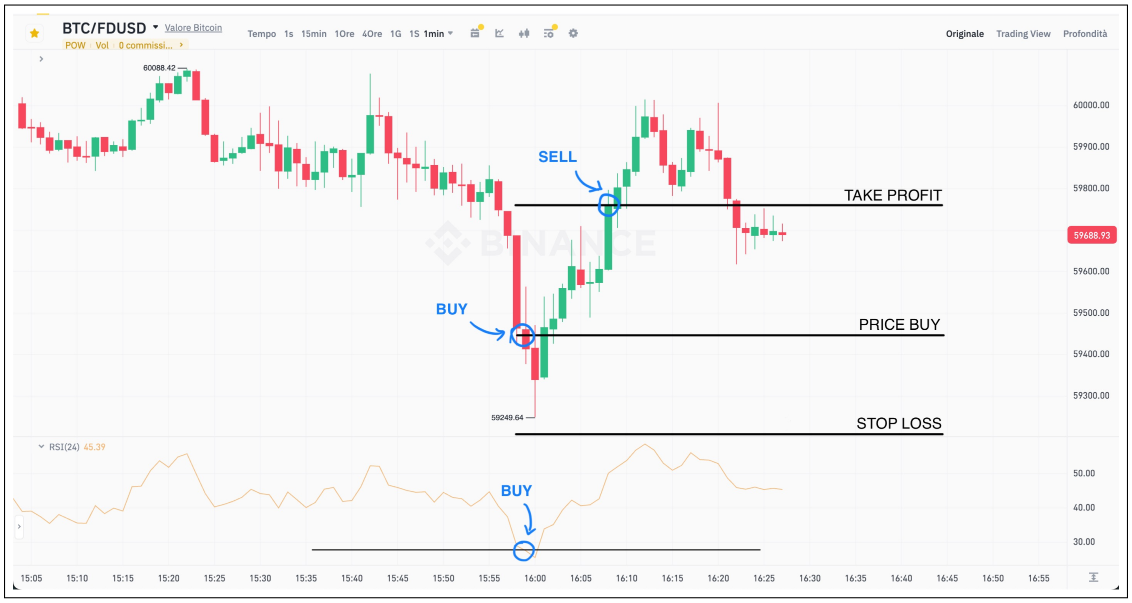

Figure 1: RSI strategy

With this somewhat functional strategy, we began testing it on large samples of data, specifically using almost 2 years of BTC price data at the seconds timeframe. To acquire this amount of data, we developed another script that ran overnight, continuously sending requests to Binance API for 1,000 lines of data each iteration (limited by the API max capacity). The data was then concatenated and divided into chunks (approximately 1 million lines each) for disk space reasons and because Pandas could not efficiently handle larger datasets. All the code of this script can be found under — directory.

At this point, we needed to test and compare the results of an optimization across more than six parameters, which was not feasible using our current approach. We tried some Montecarlo simulations on shorter periods and even attempted using Scikit-Learn and SciPy for optimization, but these efforts yielded slow and poor results.

Final Strategy

We then decided to incorporate another indicator: the EMA, which stands for Exponential Moving Average. The EMA calculates the average price by giving more weight to the more recent periods. In our case, we used two EMAs, a 100-period EMA and a 200-period EMA. By measuring the difference between them, we could gain insights into how the price is behaving. Specifically, when the 100-period EMA is higher than the 200-period EMA, it indicates that the average price over the last 100 minutes is higher than that over the last 200 minutes, suggesting a rising market trend.

Based on this, the bot continuously scans for crossovers (when the 100-period EMA crosses above the 200-period EMA) to enter a trade. We also made some adjustments to avoid high volatility periods where the EMAs keep crossing each other, leading to unpredictable results. These adjustments can be found in the `CrossingEMA` class within the `indicators_file.py` in the `code` directory.

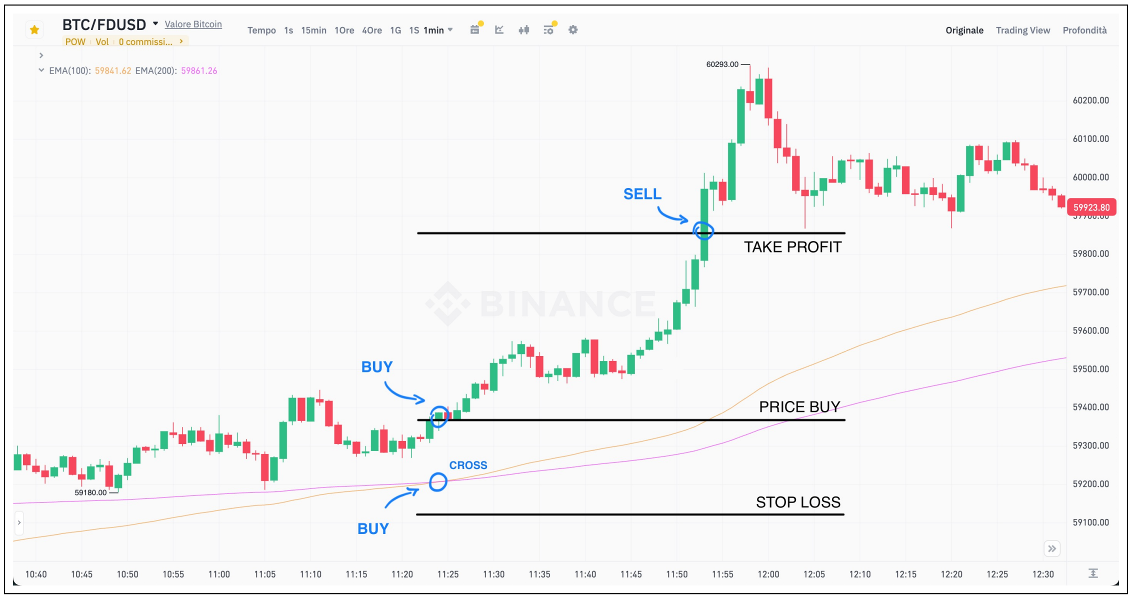

Figure 2: EMA strategy

Backtesting and R optimization

Once we had identified this promising strategy, we began our testing journey by running it over the 2-year span we had gathered. Our goal was to optimize only two parameters: the take profit and stop loss constants. These constants were then multiplied by a variable called `Kandles_Height`, which is simply the average height of the last six candles. This approach allows for an adjustable strategy that can adapt to both high and low volatility periods.

Running the strategy for the first time took more than an hour to complete. The issues were twofold: firstly, Python is known to be a slow interpreted language; secondly, the Python bot was not primarily written for efficiency but rather for handling different compositions of strategy. We had no choice but to rewrite everything in a faster language. With good C++ knowledge, we decided to translate the essential components to be operational. The most challenging part was finding a suitable and lightweight counterpart to Pandas. Another issue was the concatenation of different databases (1 million lines each) for which we utilized a buffer. All the C++ code can be found in the `Cpp` folder.

With this new implementation, we drastically reduced the testing time, going from hours to just 2 minutes for each run. We then proceeded to optimize using SciPy over a range of [0-30] (for both take profit and stop loss constants) but with no great success. In fact, the output of each test, being the final balance after 2 years of trading, did not produce a "nice 3D surface" thus compromising the gradient algorithm behind the SciPy optimization.

We then started sampling the $[0-30] \times [0-30]$ square and collecting the outputs in the z-axis on two different datasets, one during a downtrend (looking at day timeframe graphs), and the other during an uptrend. Once we had enough data, we moved to R-Studio for some linear interpolation to find the surface we were looking for. We tried many different degrees of polynomial and settled on *fourth grade* for the downtrend and *third grade* for the uptrend as they seemed the best fitting ones.

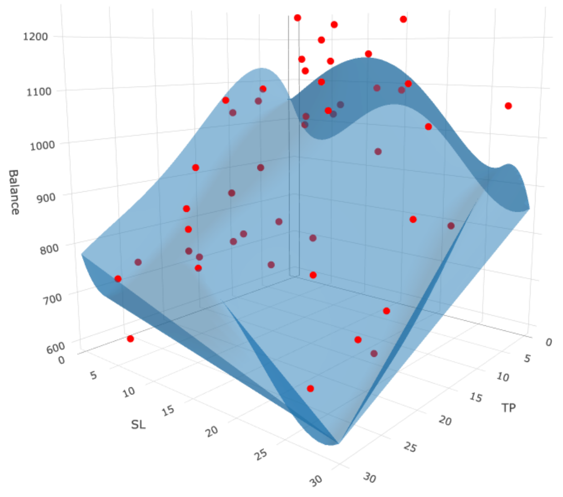

Figure 3: Surface during the downtrend

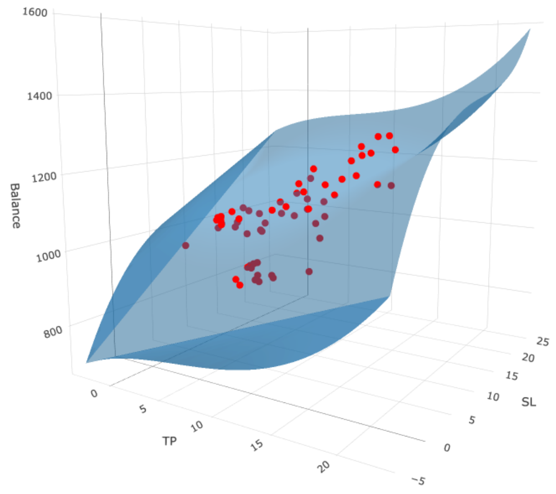

Figure 4: Surface during the uptrend

We then summed the two surfaces, giving the same weight to each trend. Once done, we chose the local maximum of the resulting surface (red point in the picture below) to be the best value for our estimation. All the R code is inside the `R` folder. I will not delve into explaining why this is the correct choice since it is outside the scope of this report.

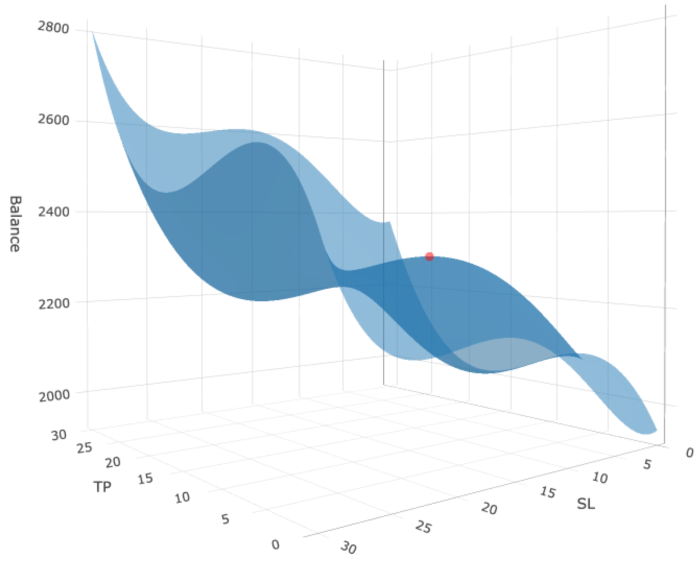

Figure 5: Summed surface showing the local maximum (red dot)

Maximum Behavior Analysis

Maximum at:
TP (Take Profit): `6.380509`
SL (Stop Loss): `16.37428`
Expected final Balance: `1156.61`

Under what we have estimated, we should expect a 15.661% gain in a year span using these two coefficients.

Results and Conclusion

The results we obtained are promising, yet it is crucial to consider a significant assumption: our testing assumed a commission-free trading environment, which is not typically the case. However, in a "subscription-based trading platform," it is still possible to profit using this strategy. As a final note, we had the bot running on our VPS during the free BTC/FUSD transaction period offered by Binance and it performed exceptionally well. We continue to run the bot, though it may now be operating with different currencies and might be trading with real money or in a simulated environment. To see the real-time data, visit rikygirg.com/home, then register to gain full access to trade history, current EMA values, and more.