Automatic trading for a time series

GOAL:

the goal of the project is to see if it's possible to create a trading bot capable of autonomously buying/selling every kind of cryptocurrency/stock/other in an automatic way. The idea is to use a mixed approach comprising classical technical analysis (moving averages) and a new intuition of mine, involving machine learnig, to detect the right moment in which to buy or sell a title.

HOW IT WORKS:

As said, the final resulti s the combination of two approaches:

1. **MACHINE LEARNIG**: given a series of values, the algorithm takes into accout every possible sequence of length N.

   (es: sequence = 1,2,3,4,5,6 and N = 4 → (1,2,3,4 ; 2,3,4,5 ; 3,4,5,6))

   All of these sequences are then normalized by simply dividing them by their average value in order to have them comparable and centered around 1. Done this, the generated sequences of length N are used as a input to generate a K-Means Clustering model having 3 clusters. One for the ascending sequences, one for the descending ones and the last for the stable ones. The precision in identifying trends in very high using this technique.

2. **TECHNICAL ANALYSIS**: once the trend has been identified using the machine learning model, the two moveing averages are used to confirm the process. They give a BUY signeal if the short average is above the long average and vice versa for the SELL signal. This because the short time moving average responds faster to changes than the long time one.

THE PROBLEM:

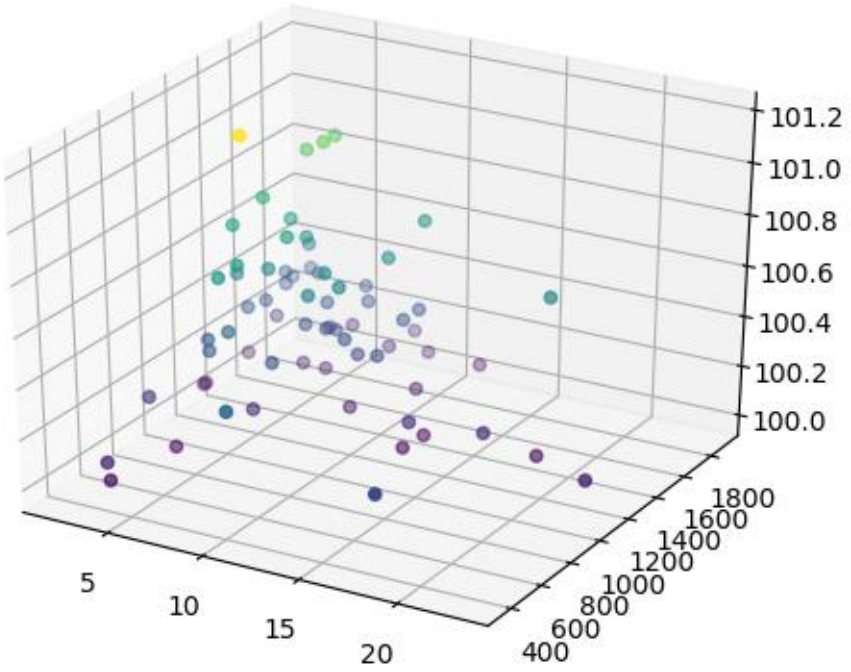
detecting the best configuration means finding 3 parameters: N, the length of the short moving average and the length of the long movign average. Being a parameters discovery problem, time complexity becomes O(n^3).

To evaluate a configuration (set of 3 parameters), the code exploits the simulator; it essentially applies the parameters and runs a paper simulation against the historical log. In the end it evaluates the final value of such configuration. The initial budget is set to 100. If

the simulator reveals that the final BUDGET + PROFIT (profit is whatevere exceeds 100), I'm gaining money with the configuration, otherwise I'm loosing.

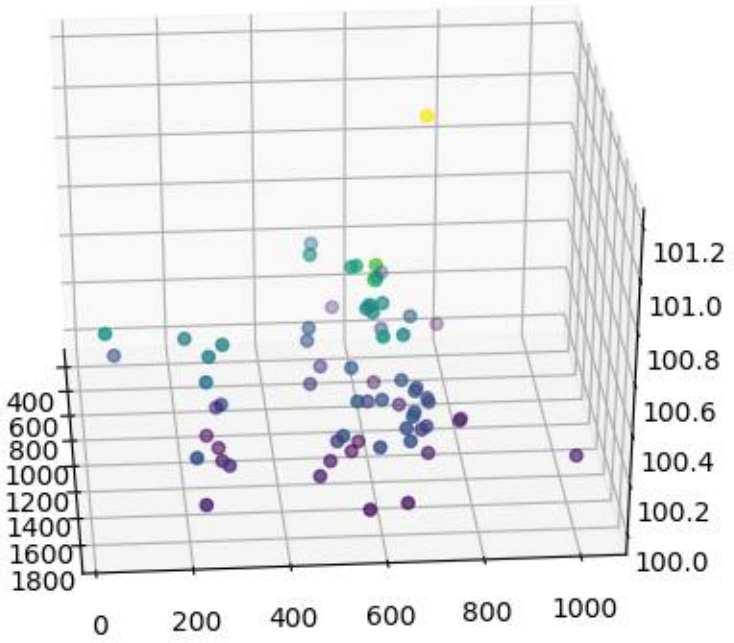
To overcome the unfeasibility of the problem, this code uses a couple of heuristics. These heuristics do not simulate every single configuration but randomly generate points. Points represent a configuration in which N is fixed and the other 2 variables are randomly generated in a finite number, (eg. 100 points for every N).

This kind of heuristic generates a function that gets plotted on a chart. By looking at the chart is then possible to see in which areas the funzioni s getting maximixed.



(The image shows how, for day 44, low values af N lead to higher values in the simulation)

A second heuristics allows to better analyze those areas and come out with the best possible configuration.



(In this case it's possible to see that for day 44 the configurations having the higher values have short ranging from 400 to 1800 while the values of long are centered around 200 or 600. In this case use the single_solver to analyze a configuration fixing N to a certain value)

THE CODE:

All of the code makes use af a log containing the values sampled every X seconds and saved in the format (a list):

Log = recordclass('Log', ['coin', 'value_ask', 'value_bid'])

Having coin as a string and the othe two values as floating point numbers.

log_completo.dat contains the history of the title from when it started to sampled while log.dat can contain a subset of that time window.

In general you may want to train your model on a subset of the whole history and then test it one the entire period.

- **Functions**: contains functions used by the scripts
- **Simulator**: accepts 3 parameters: N short long and shows you how the situation evolves in time by doing a paper simulation using your configuration
- **Chart**: shows you the trend of your data

- **Daily**_log: allows you to extract the data of a certain day. Call it passing the dai as a parameter. It generates a log.dat you can use to make experiments.
- **Function 1**: show you the relation between N and short (long in hidden). Is useful to visualize the best configurations and their value.
- **Function 2**: use it to expand the details about a configuration on the plane short-long and the corresponding value.
- **Super_solver**: it's the heuristic that generates the points
- **Single_solver**: use it to see how configurations evolve when N is fixed
- **Yea**: use it to validate your model. See if the configurations you tested on log.dat are also valid on the full log_completo.dat.