MarketRegimeDetector (MRDetect)

Detecting Stock Market regime (Bear / Bull Market)

using HMM and RNN.

A picture containing silhouette

Description automatically generated

By Emmanuel Berdugo, Asi Sheratzky, Yonatan Mirsky and Roy Toledano

Project goals:

The main objective of the project is to design a Trading Strategy that can beat the Market! It seems quite ambitious, but we believe it is achievable. We know that on the very long run (decades), investing in the Stock Market will yield higher return that any other investment vehicle. But on some shorter periods, Stock Markets can also perform really bad!

We want to build a tool that will help us to time the market.

Data Extraction and exploration:

This phase was quite simple since we took our data from Yahoo Finance package.

We decided to analyze the SPY ETF price, since it reflects the price of the S&P 500 index, which is widely considered as benchmark for determining the state of the overall economy, and it is tradable as is on the stock market, so it could enable us to have a Trading Strategy ready to be used.

Chart, line chart

Description automatically generated

We added some features to the data, mainly the Moving Average for several periods (10, 30, 60 and 120 days) and the historical (realized) volatility for the same periods. The formula for the realized volatility is as follows:

Text

Description automatically generated

We intend to use several “observations” to learn our “Market Regime States” with our HMM, and the volatility is some important observed feature. We know that bear markets are correlated with high volatility, so we would like to help our model to predict the relevant market regime by providing it with this information.

We decided to discard the first 10 years of the data, because we suspect the market behavior is a changing business, and it might be variable with time. Moreover, we kept 2 years of the most recent data for the test set, to check the validity of our model. So we have roughly 16 years of training set.

Model Baseline Building:

We chose eventually to use the Return and the 10-days Historical Volatility, since it gave the best results.

As a metric, we chose the Z-score of the daily returns. The calculation is as follows:

Z-score = mean\_of\_returns / (std\_of\_returns / sqrt(nb\_of\_days))

This is a quite frequent profitability measure in trading since we want to maximize the return of the strategy but to minimize the Standard deviation. A strategy very profitable but with a high standard deviation might be profitable only by chance… A Z-score of 3 or higher is a proof the strategy is really valuable.

Graphical user interface

Description automatically generated

We trained our model to detect between 3 hidden states, and we will now try to have a look at that he did…

Chart, line chart

Description automatically generated

It seems the predictions on the train set are quite good, in this case we see that the state “1” seems to be the loser\_state. The “0” state seems to be the winner\_state, while the “2” state seems winning also, but with a higher volatility. Let’s check.

Graphical user interface, text, application, Word

Description automatically generated

The results are as we expected.

Let’s have a look at our trading strategy now. It will be very simple: when we are in the loser\_state, we will be out of the market (selling if we were long – doing nothing if we had no position) and we will be long in the other cases (buying if we had no position – doing nothing if we were already long). Then we compare the graph of the value of our position (rebased at the same index).

Chart

Description automatically generated

We see here a clear advantage for the HMM Regime detection based strategy! We got out of the market during the bear market phase of the Subprime crisis, and got back to the long position at time. Overall we achieved roughly 70% higher return on the whole period, and, even more important, got a Z-score almost twice as good! This means our strategy seems valuable!

But we need now to check the results on the test set…

Graphical user interface

Description automatically generated with medium confidence

Chart, line chart, scatter chart

Description automatically generated

It looks good! Let’s have a look at our trading strategy…

Chart

Description automatically generated

Well… We need to say this is far above our expectations! It’s hard to believe these predictions were done step by step by the model! The model managed to avoid all the bear periods! Overall it achieves a performance 54% higher (77.63% instead of 50.45%), with a very high Z-score (3.838 instead of 1.328)! No doubt that even these results are particularly valuable for trading! But we would like to achieve better…

Next Steps:

We want to be sure of the validity of our model, so we will apply it to other indices to check if it works

We want to try other methods, based on Deep-Learning (RNN, LSTM, GRU, Transformers?) to try to achieve better results.

Phase 2…

Back to earth… Our initial results seemed a little bit too good to be true, so we decided to investigate further. It turns out we provided all the test data to make the prediction, believing the HMM Model would take the test points **one-by-one** and predicting one-by-one… But is it really the case? Or does the model consider all the points then try to find the relevant regime changes on the whole test set?

We ran the prediction again, but this time by giving the point one by one and predicting each point with all the points before (all the train set + all the points in the test set predicted earlier by the HMM). For each prediction we took only the predicted point, and we put all the predicted points together, and here is the result…

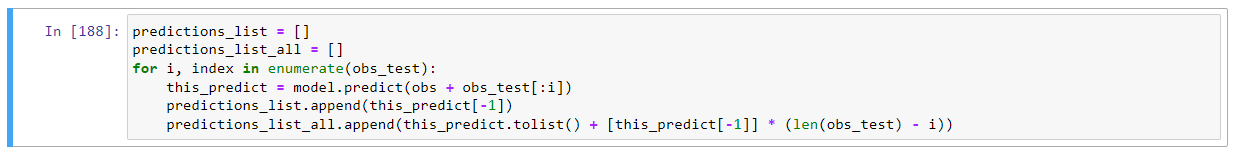
Chart

Description automatically generated

Well… It seems less bright than previously… We did not reach a better overall trading performance with the HMM. We detected the “Corona” market crash and got out of the market on time, but unfortunately it took too much time to get back to the market.

Nevertheless, we can observe some very interesting result: the Z-Score of our Strategy using HMM is far better than the Long Only strategy! Even if we did not increase the overall performance, we managed to reduce the volatility of our trading, resulting in a very significant increase in the Z-Score! For investors willing to take less risks on their portfolio, even at the price of a slight decrease in the Profit, this strategy can be used.

We would like to check further into our initial mistake, and to know if when we use the model.predict() on data of our train set + an increasing number of days we get the exact same path or not, on the train set and on the points of the test set. So here is the code we ran:



In the “predictions\_list” we kept only the last predicted point, and in the “predictions\_list\_all” we kept a copy of the full path of each prediction, padding it with the same last value till the end (to be able to compare the outputs further).

Then we drew a heatmap of all the predictions\_paths for each one of the 503 points of our obs\_test – out of sample. And here is the result:

Chart, bar chart

Description automatically generated

It sounds Interesting! Let’s check separately train and test set.

Background pattern

Description automatically generated

What we see on this graph is the path predicted for the train set only (up to point 4,047) for all the subsets of the test set we add. And we can see that they are all the same.

Let’s check the same thing but on the test set…

A picture containing background pattern

Description automatically generated

This figure shows the path to the end of the test set for each subset. For example, the top line contains only one color (state = 1) since this single prediction is continued to the end of the series. And the bottom line contains the “final” path, for which we predicted 503 times the next step. This explains the specific pattern of this graph.

But when we have a closer look on this graph, we detect 3 anomalies! Around the points 126, 357 and 386, we see that the prediction for time “t” will eventually be overridden by a later prediction, based on a dataset that will contain few more points…

So it looks like this method is still not perfectly consistent with itself, since it will “correct” backward predictions once it get more data…