Data Science

Sergey Malchevskiy Follow

Sep 10, 2019 · 6 min read · Listen



# Unsupervised Learning to Market Behavior Forecasting

This article describes the technique that forecasts the market behavior. The second part demonstrates the application of the approach in a trading strategy.


## Introduction

The market data is a sequence called time series. Usually, researchers use only price data (or asset returns) to cre e next price value,

movement direction, or other output. I think the better way is to use more data for that. The idea is try to combine versatile market conditions (volatility, volumes, price changes, and etc.)

The first type of potential features are the various derivatives of price data. The second type is the set of the volume derivatives.

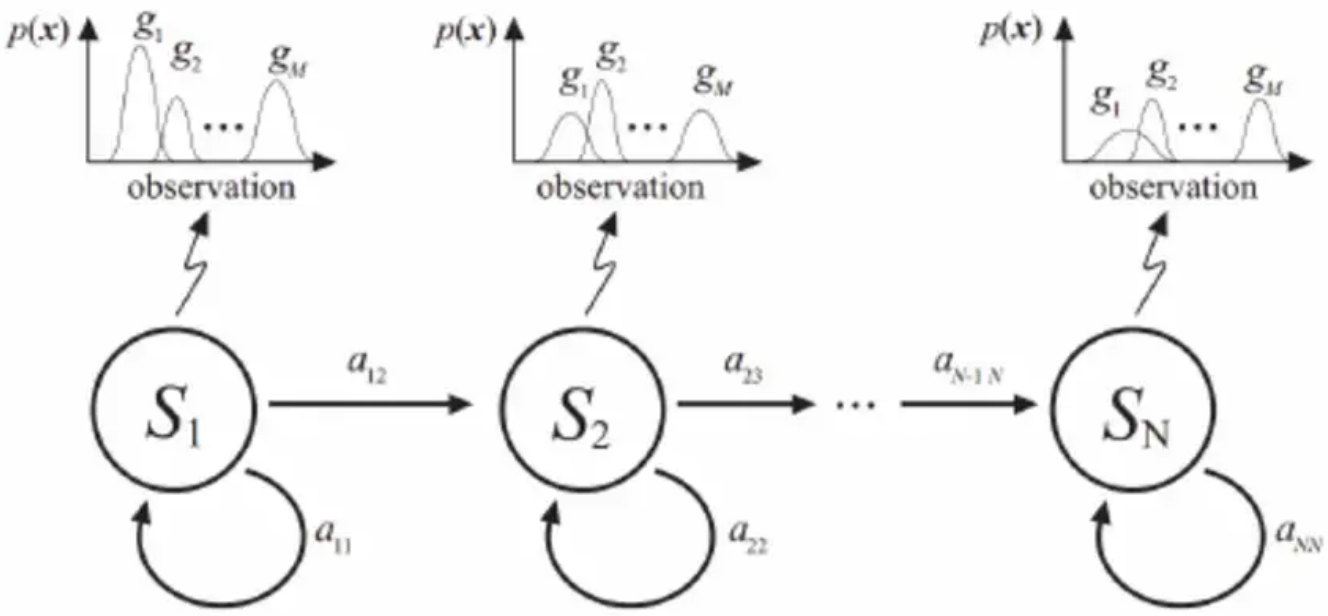
> These features will describe the current market condition more complex than raw market data or simple returns.

You will see these features in the next part of the article. As for the modeling, we will use Hidden Markov Model.

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (i.e. hidden) states. Observed data is our market features, hidden states are our market behavior.

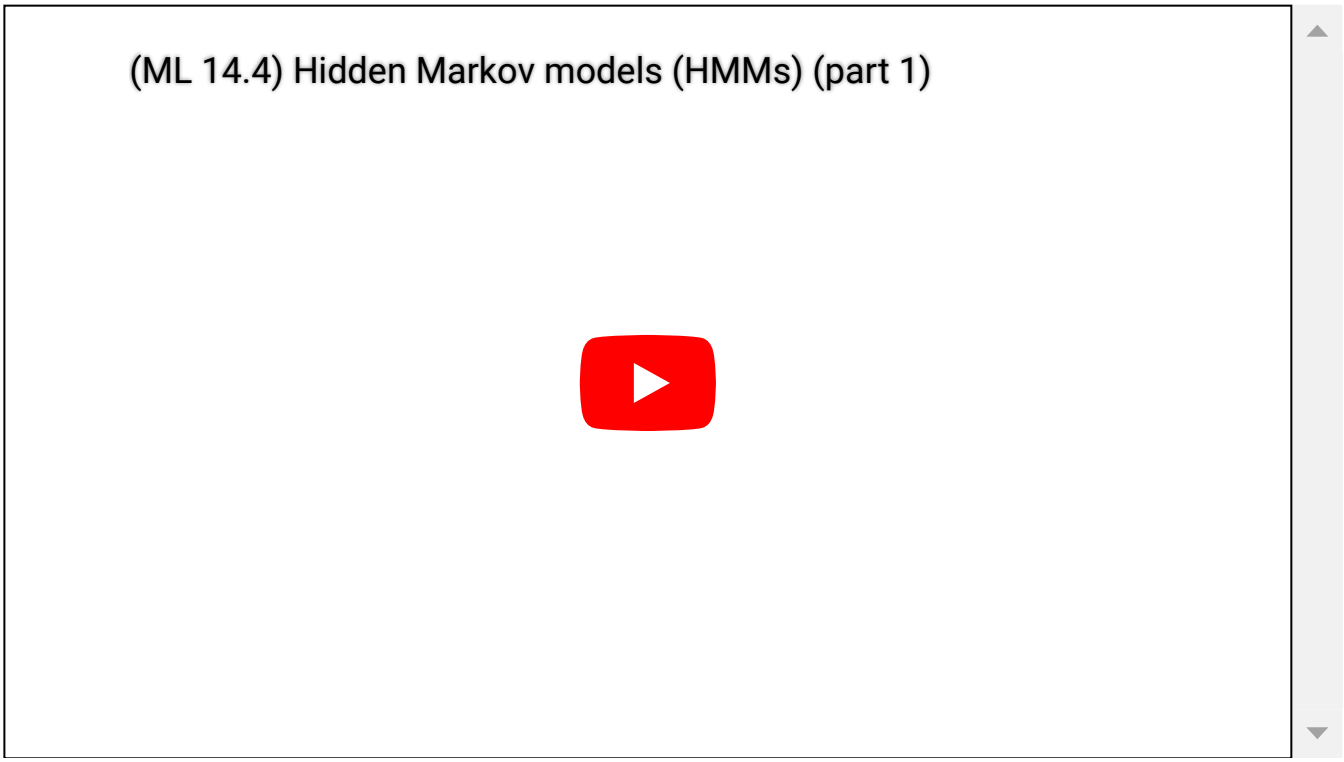
> Our goal is to interpret the hidden state after the modeling, and create the trading strategy based on this knowledge.

The basic figure of Hidden Markov Model looks like this



Hidden Markov Model

This article is practice-oriented. For more information you read the introduction to Hidden Markov Model in this article by Tomer Amit. Also, I recommend to start with this video




## Feature Engineering and Modeling

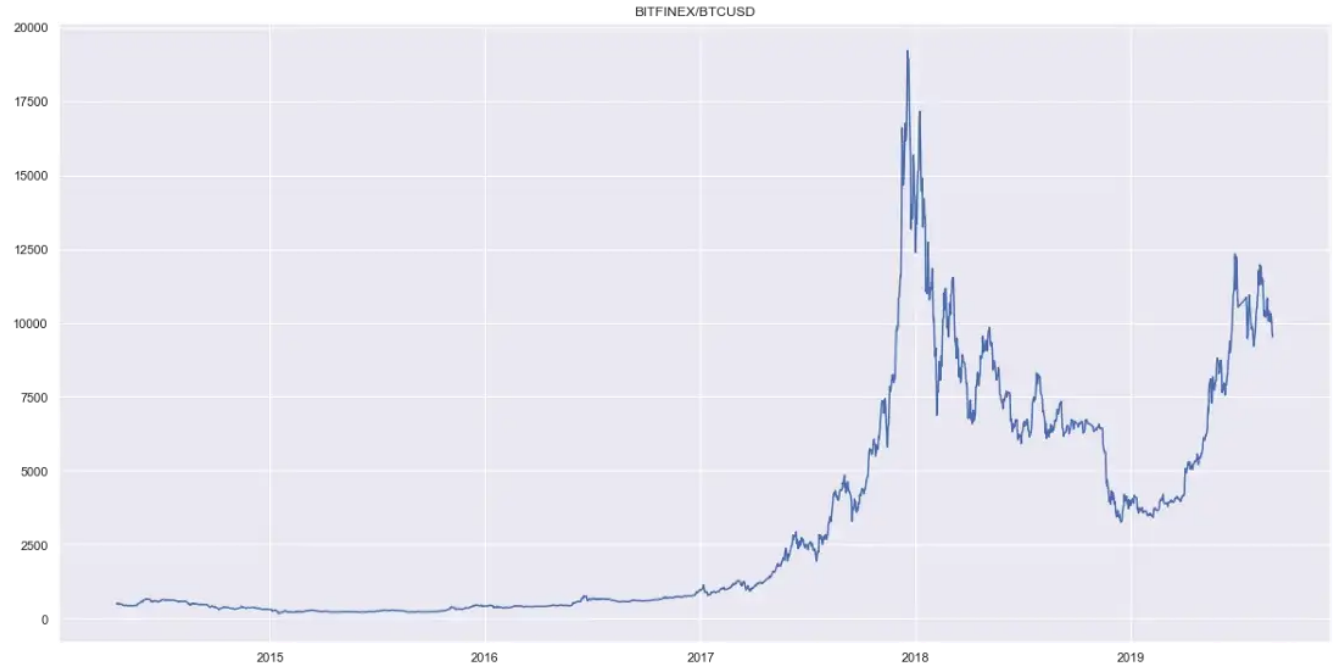
I think combing of code and explanation is a good approach to dive into the research. Let's start coding.

Our libraries

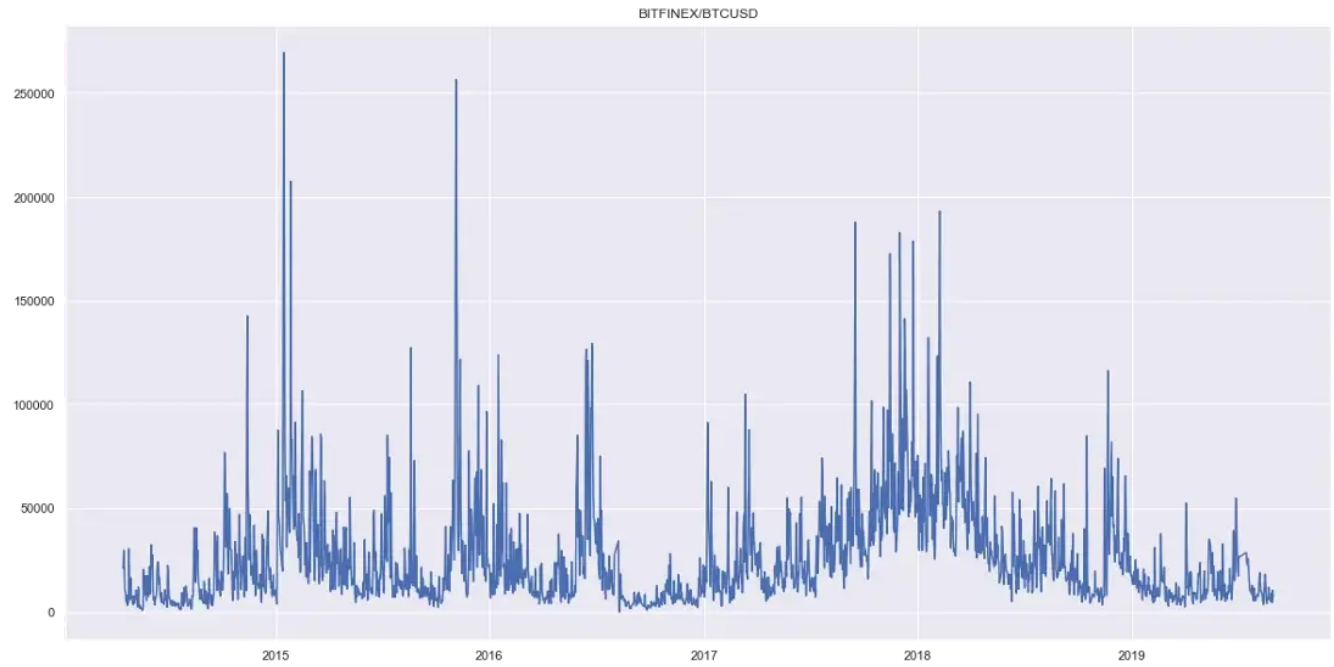
This code downloads data for BTC/USD from Quandl

Then we can to plot the price and volume data

After that we are getting this figure




Price for BTC/USD from 01/01/2014

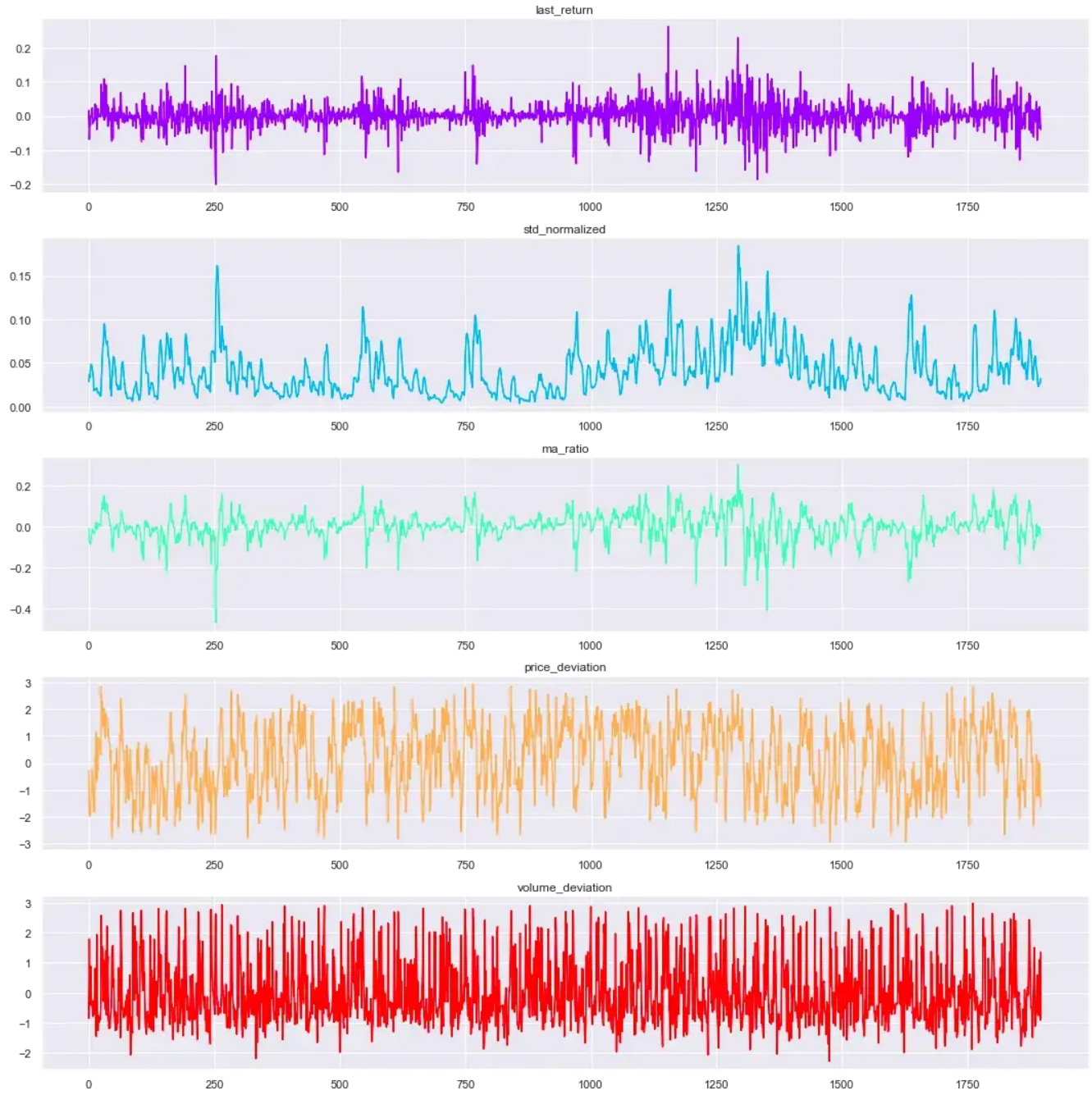



Volume for BTC/USD from 01/01/2014

Now we are ready for coding the feature engineering and modeling functions.

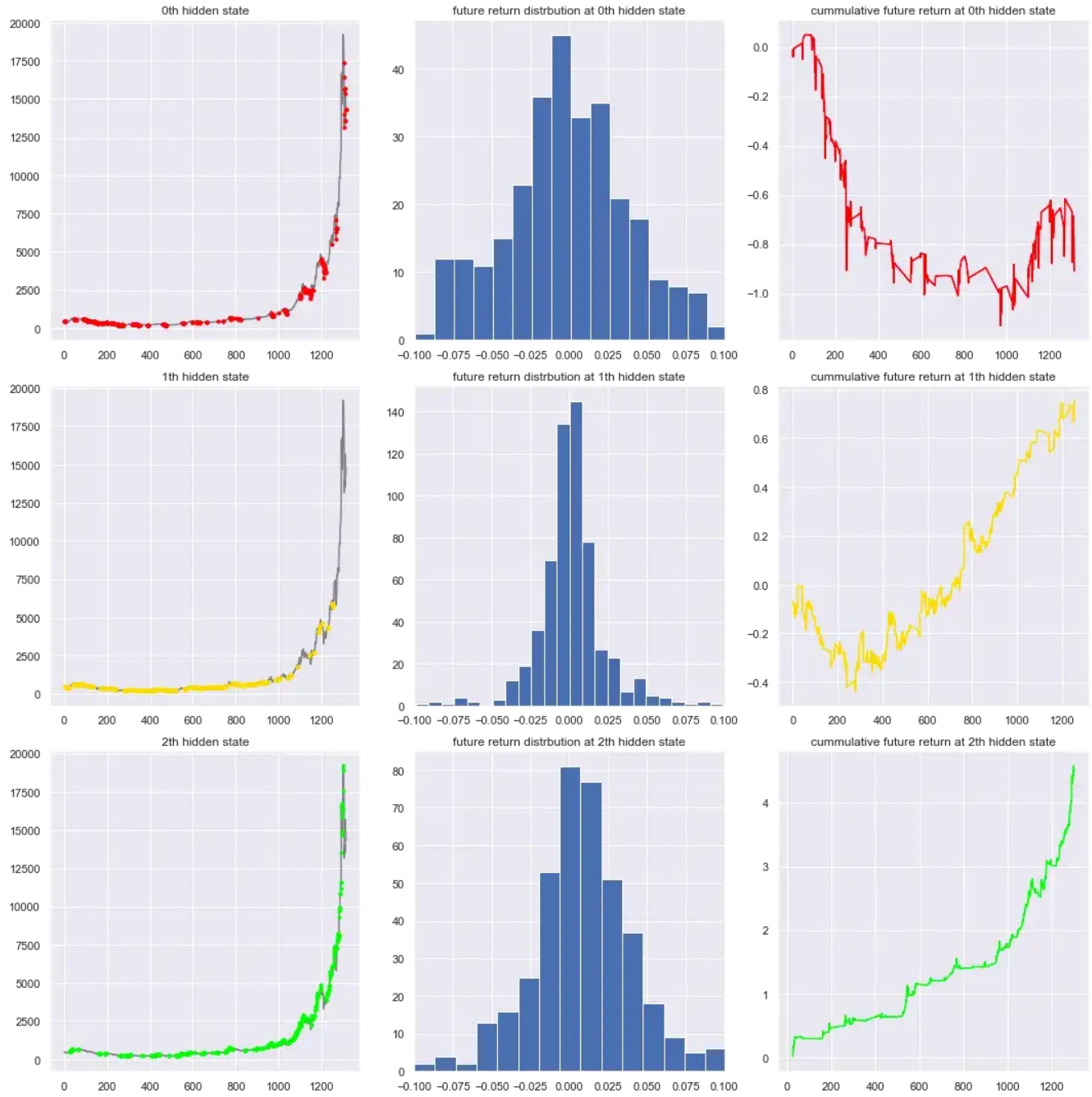
Let’s split off the train period as a period before 01/01/2018. The next code runs the feature engineering and visualizes it.

After that, we are getting the five new time series and the trained model.




Feature sequences

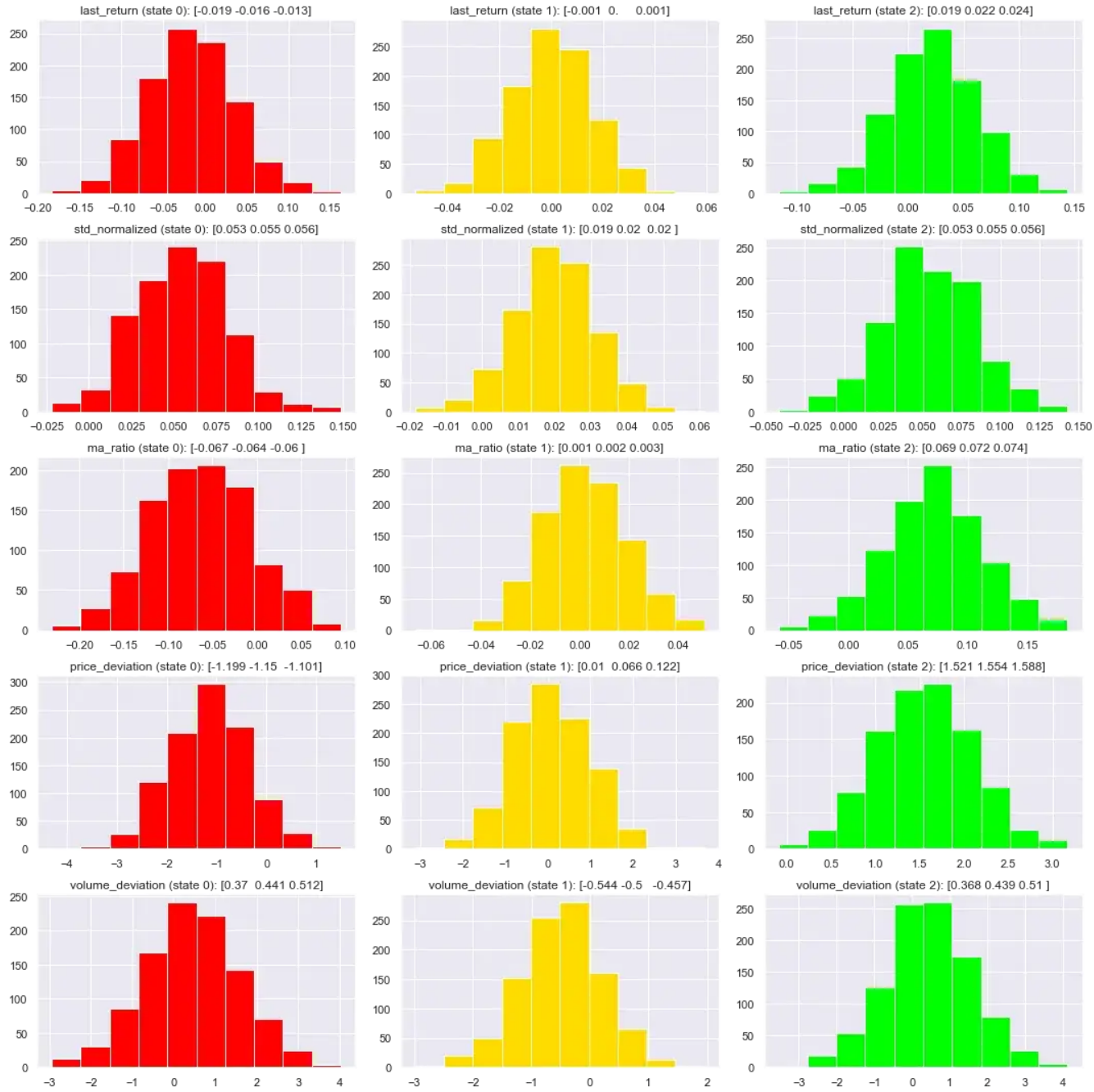
In the code above, we also created the *future_return* column that shifted by one lag for *last_return*. This is the **first key** to understand the hidden states. Let's plot this value as a cumulative sum by each state.




Future return cumulative sum for each state

As we see, state #0 has the tendency to downside direction. State #1 doesn't have a clear tendency. The last state #2 has a strong tendency to upside direction. This simple trick with a cumulative sum *future_return* allows us to understand how each state corresponds to the next price movement.

The **second key** is to research each state by features. After that, we can relate these two events (future movement and current condition). Let's code simulation and visualization for features of each state.




Feature distributions for each state

Now you can see how each state describes the current state. E.g., states #0 and #2 have a high volume deviation, it means that these states often presented on high volume, and state #1 on low one. Also, states #0 and #2 often presented on high volatility.

The interesting fact is state #0 has the low values for *last_return* and *ma_ratio*. Probably, *the state #0* corresponds to downside current condition (at the present). The backward situation is for state #2.

The interpretation of that two conclusions is

If the market has the current state #0 we have mostly downside market condition (second key) at the current situation, and this tendency will go on (first key).

If the market has the current state #1 we have the uncertainty for the tendency.

If the market has the current state #2 we have mostly upside market condition (second key) at the current situation, and this tendency will go on (first key).

The next line of code saves the trained model into the file.

## Application

Let's try to create the trading strategy based on this knowledge. The strategy we should test since 01/01/2018, because this period out of the sample.

> The logic is simple: short when state #0, no position when state #1, long when state #2.

Our strategy will be implemented using Catalyst framework. In this post, I demonstrated a quick introduction to Catalyst. The strategy will be contained in

separated py-file. Let's code basic functions and include the libraries

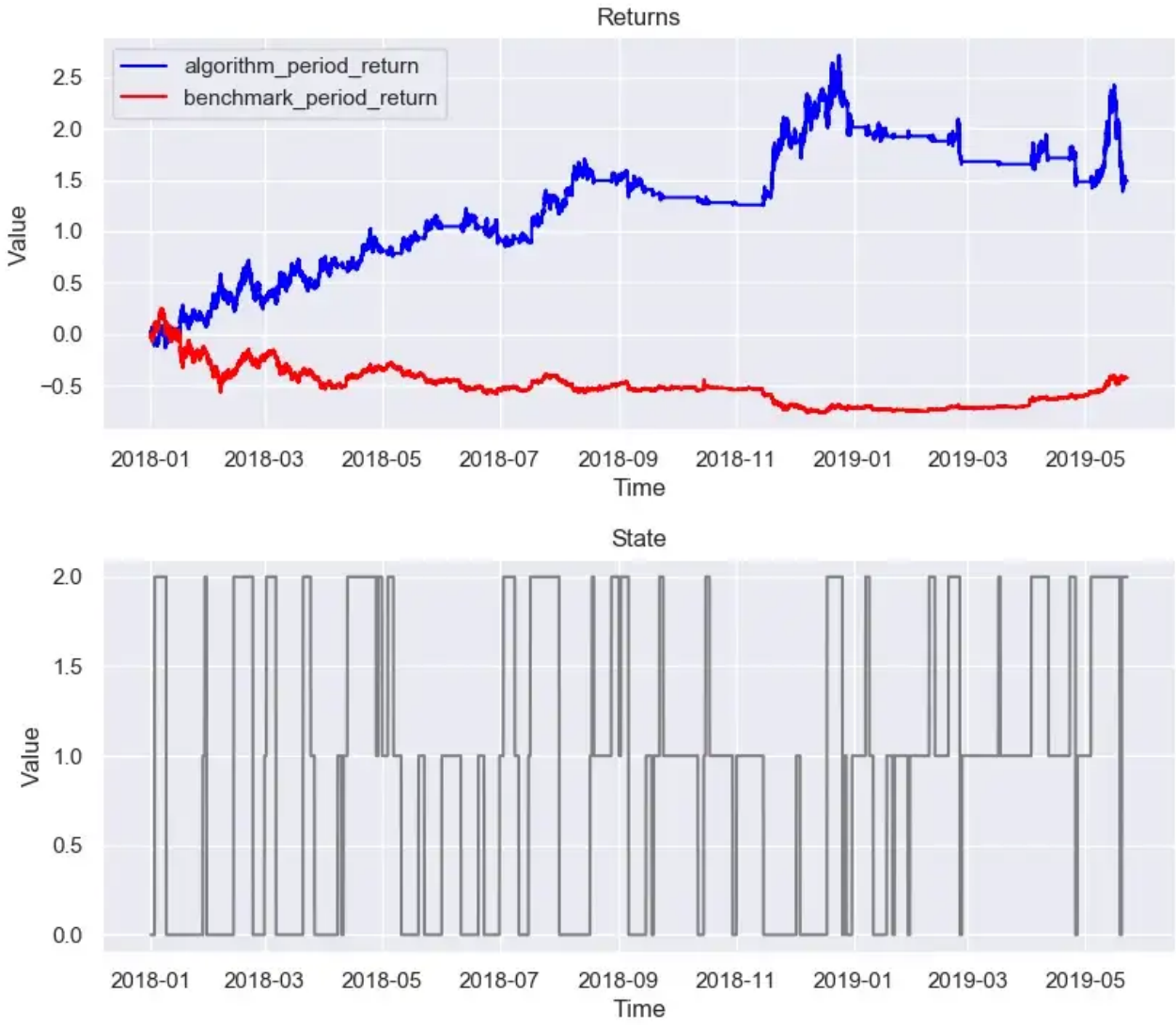
The main parameters, model loading are contained in the *initialize* function

The basic logic contained in handle_data function. This function runs every minute. The main activities are getting data, feature creating, market behavior estimating, and position management.

The last function is additional. We plot the figures and print the result.

## Let’s run the strategy

As we see, the suggested algorithm straight beats the benchmark. It seems the strategy tries to catch the trend and following it. The bad condition for the strategy is no trend period.




Backtesting result

> *Total return: 1.4889661611137708*
>
> *Sortino coef: 1.8800527725096972*
>
> *Max drawdown: -0.30508556497453887*
>
> *alpha: 0.5672854146797404*
>
> *beta: -0.1541654082608784*

Alpha is positive, beta is very close to 0 (see this post for the definition of these criteria). The drawdown is too high, but it much lower than drawdown of the benchmark.

## Conclusion

1. Suggested the approach based on the versatile price and volume features as sequences.
2. Carried out the interpretation of hidden states of the model.
3. Built the model using Hidden Markov Model on data for 4 years.
4. Created the simple trading strategy that tested on out of the sample data (1.5 years) without retraining, commission and slippage included.
5. The strategy beats the buy & hold benchmark, and it has positive alpha and beta is close to 0.
6. The research artifacts are uploaded to GitHub



**The ways how to improve the achieved result:**

1. Add new features to the model.
2. Experiment with the window length.
3. Build the model with different number of hidden states.
4. Try to formulate the new interpretations for the hidden state and rules for employing in the strategy.
5. Create the portfolio based on a few assets.
6. Add the simple trading rules such as take profit, stop loss, and etc.

These improvements can help to get more sophisticated strategy result: reduce the drawdown to 15–20%, increase the alpha and sortino, increase the capacity.

If you like this sort of applications you can read this my article based on similar approach.

Best regards,

Sergey

***Note from Towards Data Science's editors:*** *While we allow independent authors to publish articles in accordance with our rules and guidelines, we do not endorse each author's contribution. You should not rely on an author's works without seeking professional advice. See our Reader Terms for details.*

Stock Market Data Science Python Machine Learning Cryptocurrency