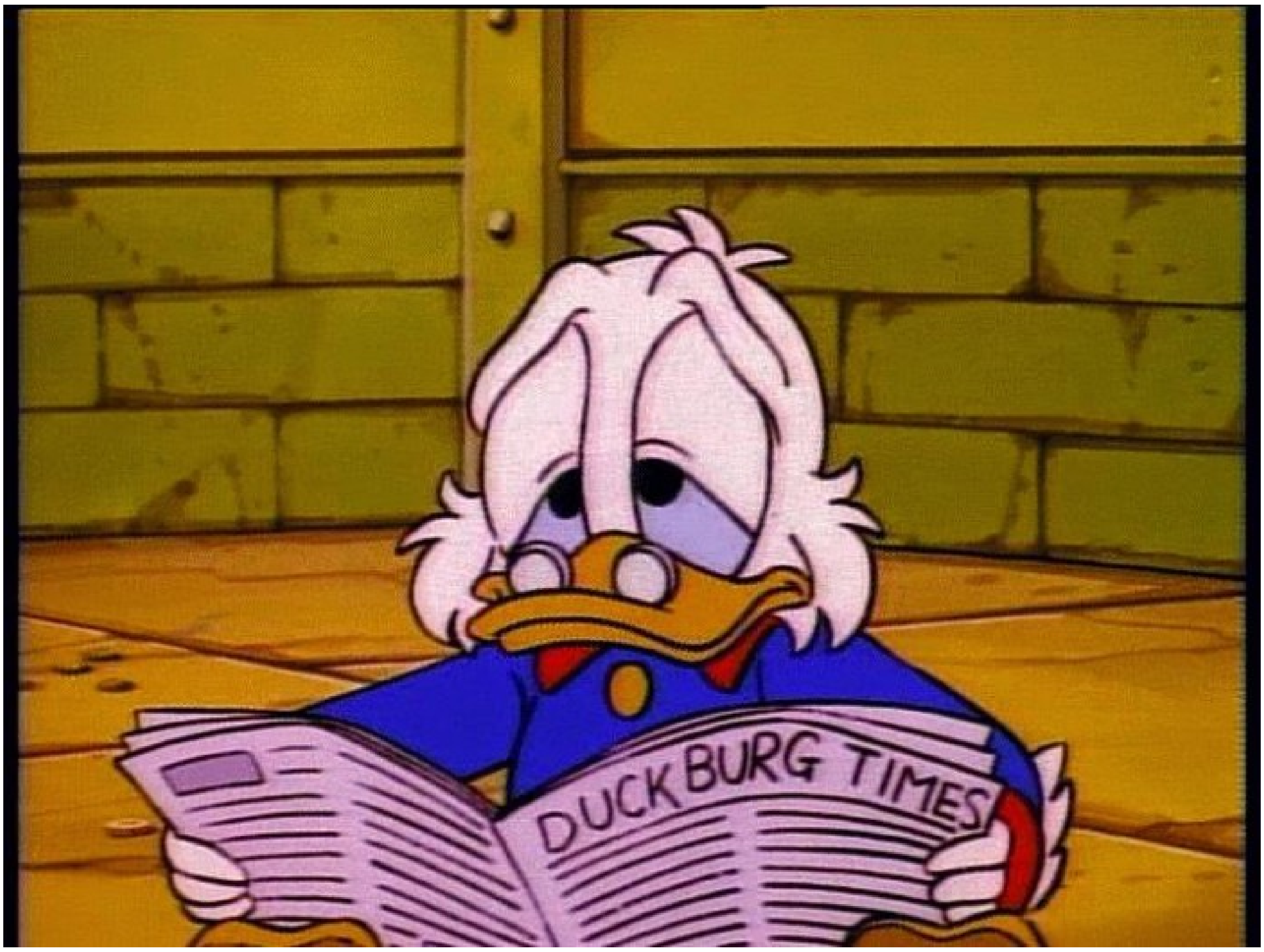
AI for algorithmic trading: 7 mistakes that could make me broke



Hi everyone again! It's been a long since my last post about machine learning for algorithmic trading and I had some reasons for it. After I could show some rather successful results in forecasting assets prices using neural networks I received interesting offers from different institutions: starting from banks and ending family funds and individual traders that were expecting to have some fruitful earnings using AI. I already had some strategies properly backtested (as I thought), so I was happy to help them. Meanwhile I also talked to several experts in the area and studied some literature and things appeared to be not that easy as someone can think. Today I want to share with you lessons I've learnt from professionals, some literature and, the most important, own experience, that can potentially save you a lot of money if you're trying to incorporate AI in your trading process.

First of all check my other articles please:

1. Simple time series forecasting (and mistakes done)
2. Correct 1D time series forecasting + backtesting
3. Multivariate time series forecasting
4. Volatility forecasting and custom losses
5. Multitask and multimodal learning
6. Hyperparameters optimization
7. Enhancing classical strategies with neural nets
8. Probabilistic programming and Pyro forecasts
9. Backtesting with Pandas

to be probably impressed with possibilities… and prepare to get discouraged :)

Current framework

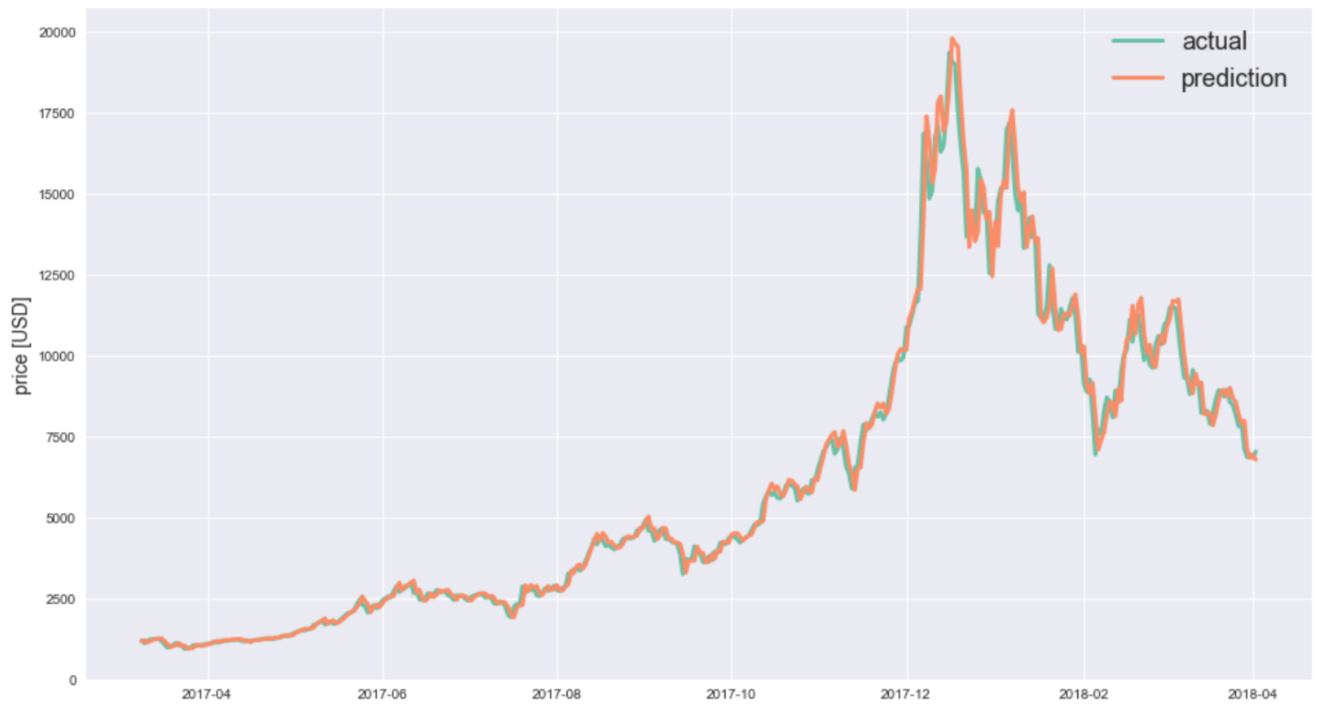
As you can see, my whole blog is concentrated on solving the following machine learning problem: having a window of past N observations predict N+t one, where N can be 30 days of bars (or other variables as news headlines) and t is forecasting horizon (can vary depending on your strategy). What we have done **correctly** here?

1. **Feature preparation and normalization**
   Indeed, we have prepared our windows correctly: no look-ahead bias, normalization done for each window separately (not to mix different regimes etc)

2. **Different task selection**
   We also have played with different forecasting objectives: on the high level they were both classification (binary movement "up" or "down") and regression — returns forecasting, volatility forecasting etc.

3. **Backtesting**
   Training neural nets is cool, but it's not the end of the process. We performed simple, but informative backtesting to show that if we would use these predictions to long/short assets we could make some profits (what's important, better than some benchmark)

4. **Avoided classical overfitting**
   The main challenge in "normal" machine learning is to generalize our model to perform well on unseen data. To do this we use train/dev/test set splits of our initial data and do some cross validation to estimate how model behaves with changes of data distribution. We have shown, that neural networks can perform well on out of sample data (but wait for tricky details). On the image below is something what I call "classical" overfitting for financial time series, which looks as perfect forecast, but basically it just predicts the last observation :)



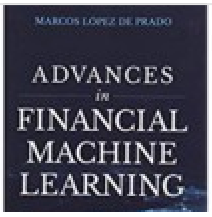
https://hackernoon.com/dont-be-fooled-deceptive-cryptocurrency-price-predictions-using-deep-learning-bf27e4837151

What's missing?

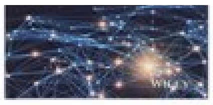
Even the above described framework is generally correct, a lot of important parts are still missing, and they are actually crucial for profitable trading. I won't talk about commission, shorting issues, liquidity, bet sizing or anything what is related actually to the strategy (I'll touch some backtesting issues, but it's needed for machine learning part). I will concentrate on problems in machine learning, dataset formation, evaluation of a model etc. It's unbelievable coincidence, but most of the solutions to the problems (at least theoretical ones) I've found in the following book:

Advances in Financial Machine Learning

Machine learning (ML) is changing virtually every aspect of our lives. Today ML algorithms accomplish tasks that until…



www.amazon.com



and I highly recommend it as well as second really useful source:

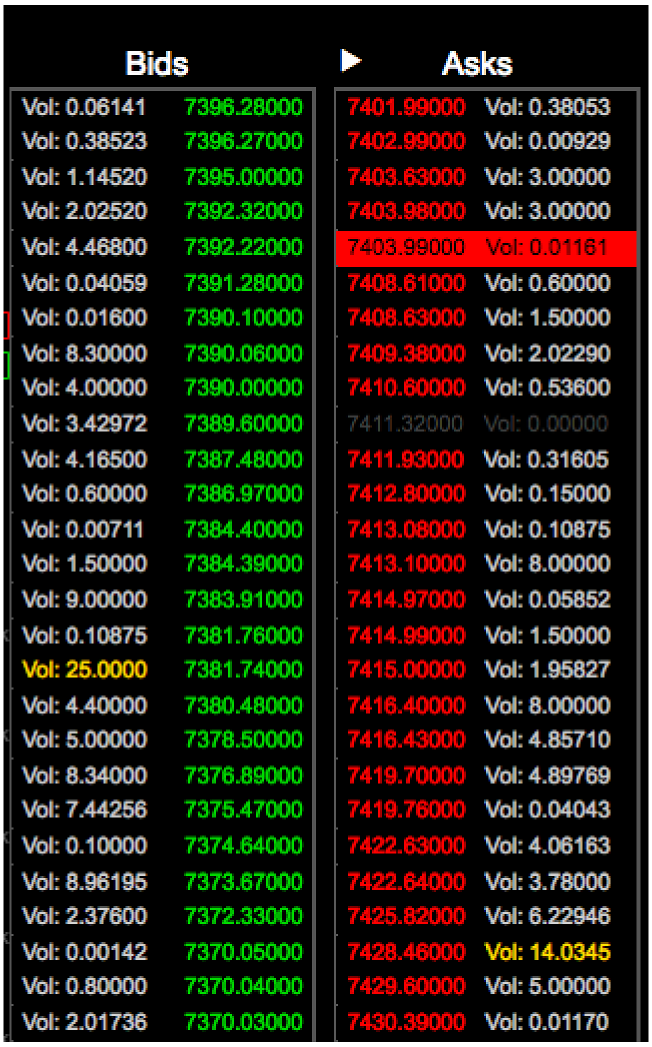
Online Lectures in Statistical and Financial Topics

Hands-on Lectures for Quantitative Finance: Teaching Scientific and Statistical Rigor Through Real Data Examples

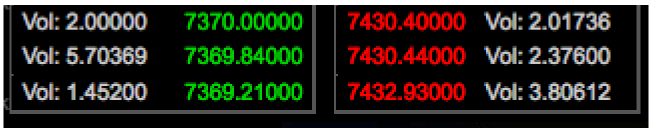
www.quantopian.com

Bars data == weak data

Most of people use well known HOLC (high, open, low, close) prices for some time period and trades volume. This information reflects not enough information about the market and what its participants are doing. We really need bids and asks from the order book to trade well — it will give us "raw" information and allows to build better features like dollar volume, bid/ask spread and others.



Bids		Asks	
Vol: 0.06141	7396.28000	7401.99000	Vol: 0.38053
Vol: 0.38523	7396.27000	7402.99000	Vol: 0.00929
Vol: 1.14520	7395.00000	7403.63000	Vol: 3.00000
Vol: 2.02520	7392.32000	7403.98000	Vol: 3.00000
Vol: 4.46800	7392.22000	7403.99000	Vol: 0.01161
Vol: 0.04059	7391.28000	7408.61000	Vol: 0.60000
Vol: 0.01600	7390.10000	7408.63000	Vol: 1.50000
Vol: 8.30000	7390.06000	7409.38000	Vol: 2.02290
Vol: 4.00000	7390.00000	7410.60000	Vol: 0.53600
Vol: 3.42972	7389.60000	7411.32000	Vol: 0.00000
Vol: 4.16500	7387.48000	7411.93000	Vol: 0.31605
Vol: 0.60000	7386.97000	7412.80000	Vol: 0.15000
Vol: 0.00711	7384.40000	7413.08000	Vol: 0.10875
Vol: 1.50000	7384.39000	7413.10000	Vol: 8.00000
Vol: 9.00000	7383.91000	7414.97000	Vol: 0.05852
Vol: 0.10875	7381.76000	7414.99000	Vol: 1.50000
Vol: 25.0000	7381.74000	7415.00000	Vol: 1.95827
Vol: 4.40000	7380.48000	7416.40000	Vol: 8.00000
Vol: 5.00000	7378.50000	7416.43000	Vol: 4.85710
Vol: 8.34000	7376.89000	7419.70000	Vol: 4.89769
Vol: 7.44256	7375.47000	7419.76000	Vol: 0.04043
Vol: 0.10000	7374.64000	7422.63000	Vol: 4.06163
Vol: 8.96195	7373.67000	7422.64000	Vol: 3.78000
Vol: 2.37600	7372.33000	7425.82000	Vol: 6.22946
Vol: 0.00142	7370.05000	7428.46000	Vol: 14.0345
Vol: 0.80000	7370.04000	7429.60000	Vol: 5.00000
Vol: 2.01736	7370.03000	7430.39000	Vol: 0.01170



Vol: 2.00000	7370.00000	7430.40000	Vol: 2.01736
Vol: 5.70369	7369.84000	7430.44000	Vol: 2.37600
Vol: 1.45200	7369.21000	7432.93000	Vol: 3.80612

https://bitcointicker.co/

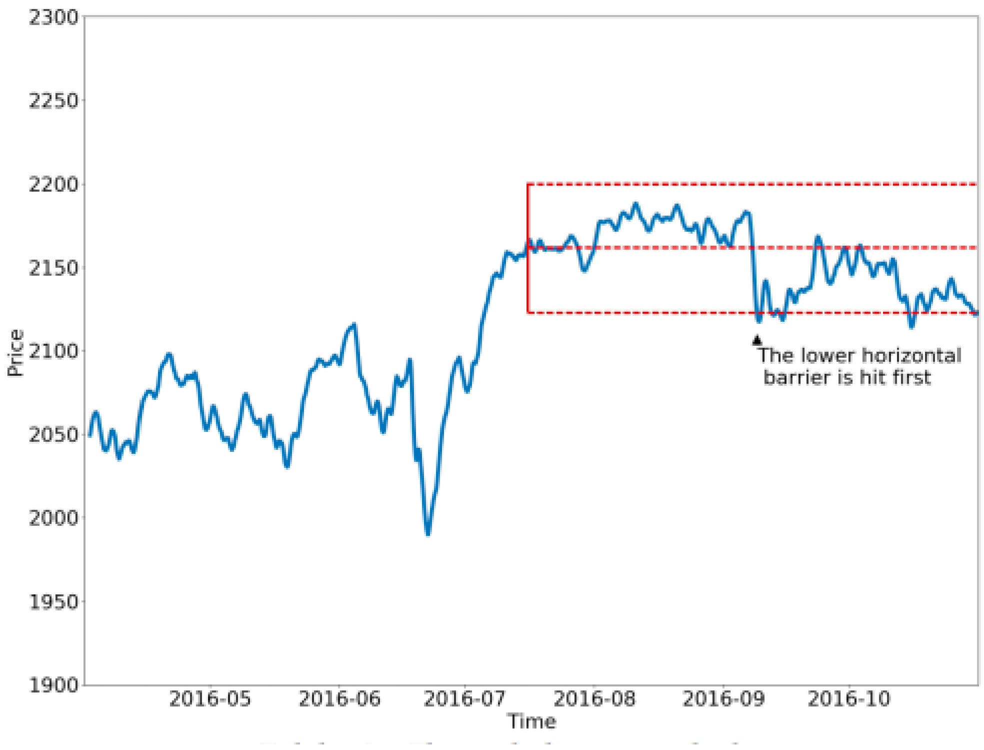
You still can build candle bars from this, but you will also get access to queues and their length, detailed information on how many people want to buy or sell particular asset (not just plain volume), tick imbalances and a lot of other interesting features, that have much more signal than just averaged noise in the bars.

Concentration on a single asset

This mistake is really bad. Most of my articles concentrate on taking some single asset, learning to forecast it for some fixed horizon and backtest the long-short equity strategy. Maybe some individual traders with $10k in the pocket really do this — they build some indicator-based strategy for some currency pair and trade it. But if we think about it for a while, it looks a lot like overfitting for this particular asset! What's the point of having a strategy that is overfitted for a single time series (and we even aren't sure that it will perform same good in the past). Hedge funds never do this. They do trade in so called universe of assets (possible with the same strategy). Portfolio is balanced to short or long these assets, or if some strategy is used to trade them it's expected to have good performance on all of them.
Moreover, when you compare your strategy performance to some benchmark (for example in case of crypto trading this benchmark can be HODL strategy) you're interested in calculation of alpha (outperformance of a benchmark) and beta (strategy risk exposure).

Fixed sized horizon forecasting

When I was preparing a dataset to train a model, each pair {x_i, y_i} was a window of N past days and price change (or direction of price movement) in some time after last date in the historical window. Let's think about it again. Some time after. Fixed time. Well, the word "fixed" in financial world is ridiculous. We can't even be sure that in some time there will be bids or asks to perform the trade! This is very serious issue, that actually ruins all our forecasting framework. To be honest, I haven't found any easy fix for this problem, just two, but they're drastic. **First solution** is stop forecasting and start executing trades, which leads us to control theory and reinforcement learning immediately. It will help us to deal with any fixed time horizon (at least to some extent), but it's a bit off the topic for now. **Second option** I found in a book and it's pretty interesting.



http://www.smallake.kr/wp-content/uploads/2018/07/SSRN-id3104816.pdf

Basically on the picture above you can see “non-fixed-time” creation of labels for the dataset. It’s called “triple barrier” and works like following: we build three barriers — one on the top, that will mean taking the profit, the bottom one as a stop loss and the last, vertical one, will mean some kind of expiration period. This labelling method allows to build much more flexible and realistic strategies based on predictions.

I.I.D.

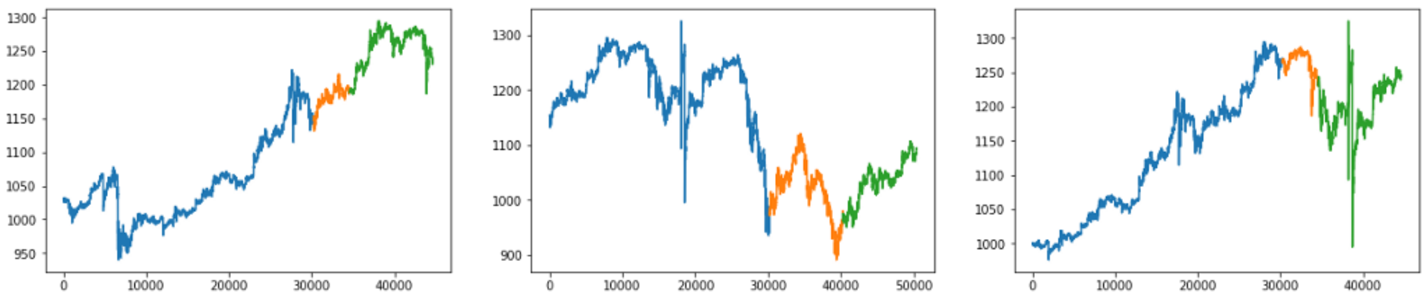
If you have read before some statistics or ML theory texts you could see this three letters i.i.d., which means **independent and identically distributed** and is used to describe some random variables. It’s true in case of most of ML applications in CV, NLP, recommender systems, even some time series analysis and signal processing… But not in the case of financial time series! Look how we prepare the data:

```
for i in range(N):
    x_i = features[i:i+WINDOW]
    y_i = (close[i+WINDOW] - open[i+WINDOW]) / open[i+WINDOW]
```

while we are iterating on *i,* we roll on a time series with some step and it occurs, that different target *ys* aren’t actually independent! Corresponding *xs* have same features, same returns inside, just in different positions. And it actually violates all our ML framework. Solutions are rather difficult and what I’ve tried by myself was kind of ineffective — while working with minute bars I took only non-overlapping windows, and this data was kind of enough, but it definitely won't be enough if you plan to work on larger timeframes. Some interesting, but rather advanced solutions I found in this amazing book again.

Validation set usefulness

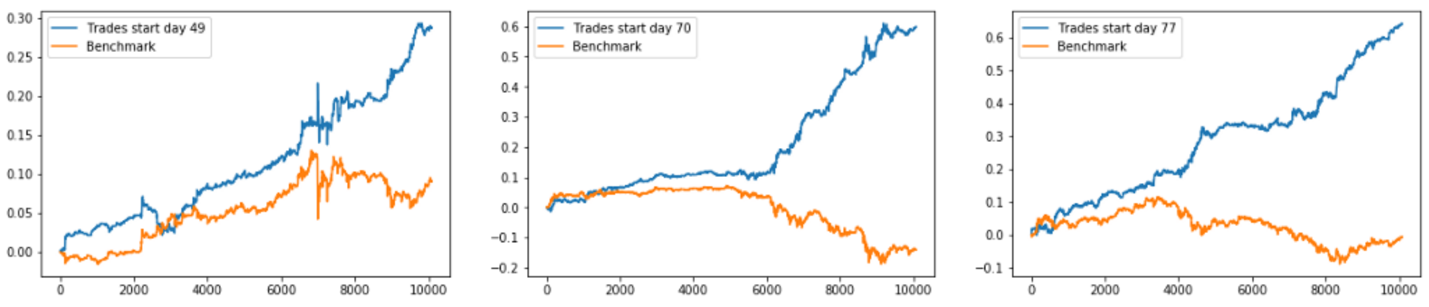
When we talk about neural networks training with, let's say, Keras, we used to pass to the *fit()* function such data samples as X_train, Y_train, X_test, Y_test, X_val, Y_val. Visually the can look like following (blue time series part is train set, orange is validation and green — test):



These splits represent a case, when we retrain a network each N days (N = 21 for example), validate performance on next 3–7 days and then if it's satisfying run it on out of sample trading period. Look reasonable… in case if train, validation and test set are similar :) for example on the first picture above there is same trend and more or less same volatility, so the neural net trained on upwards trend, validated on same trend and tested on same trend will show good results. But what can happen another month? Situation can change drastically, market behavior can be totally different and our assumptions will be totally wrong! What can be the solutions? First of all — variables analysis and selection of the ones, that **don't change** like that over time. One of the example of such features can be fundamental data or some bars patterns. Second — use **cross-validation, forward testing** and models **ensembling**, which will allow to generalize.

Backtest overfitting

Backtesting of a strategy supposed to give you an intuition about how it will perform in the future. I mean, that you believe, that if strategy showed good performance on the past data (taking into account, that there were no look ahead biases or other data processing / ML related problems) it will be more or less same profitable in the future. It's simply not true. Below you can find some equity curves from the extremely simple strategy backtests:



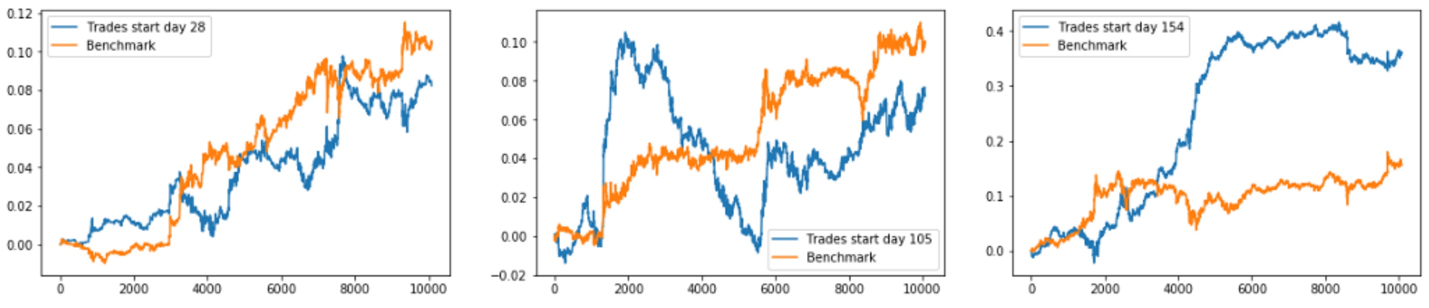
Nice looking equity curves and comparison to the benchmarks

1. QDA based on 5 minutes OHLCV bars and couple of very simple features predicts next minute change (binary classification)
2. Train and val tests are 21 / 30 days and 7 / 14 days respectively (being optimized)
3. Backtest is evaluated during next 7 days with simple long-short equity strategy.

You can find more details about this strategy below, but it outperforms benchmark almost every week. Can you trust these backtest results? Moreover, can you trust them knowing that it has all the above mentioned mistakes apart of having some poor substitute of cross-validation?

Good forecasting != good trading

Let's talk a bit more about the strategy backtests from the last paragraph. What can we understand more from them? For example let's check if the strategy returns really correlate with accuracy of the ML algorithm?



F1 score = 0.68, F1 score = 0.62, F1 score = 0.54

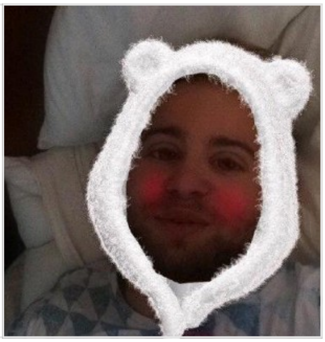
On the graphs above there are three equity curves and related F1 scores. First of all we see that on the graphs where benchmark outperforms our strategy F1 score is 0.68 and 0.62 and in the opposite situation F1 score is slightly below 55%. If you check the Jupyter Notebook later you can see that the best equities we have with F1 score around 55%, not more. To be more precise, the correlation coefficient between Sharpe ratio improvement (difference between sharpe ratio of our strategy and a benchmark) and F1 score is **-0.425**. Amazing, right? It's negative! What can we conclude? If the experiment was set up wrong and data was really random and not prepared, than everything else is random as well. Trash in — trash out.

Good looking, but bad in fact strategy example

Rachnog/Deep-Trading

Deep-Trading - Algorithmic trading with deep learning experiments

github.com



In the notebook you can find an example of something that looks nice, but in fact is not. We chose a single asset, data preparation has issues with i.i.d. samples and fixed horizon forecasting, features are not much better than bars, cross-validation is very simple walk-forward process and we definitely overfitted with a backtest, because our algorithm accuracy is even not correlated with returns.

Conclusion

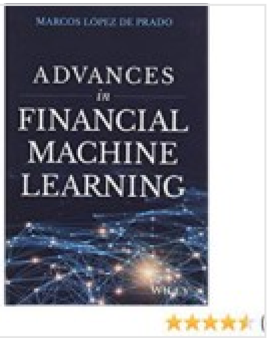
Financial machine learning is different and difficult. I am sure I am still missing a lot, and resources I used are not covering all the problems, but we can see, that simply playing with Keras neural nets is definitely not enough. It can be a nice exercise to avoid overfitting or nice proof of concept that your data, your features and your algorithm have some general predictive value, but it won't make you money. I am 99.99% sure. The 0.01% left I leave for the luck factor, but… I don't think hedge fund managers will invest in this alpha :)

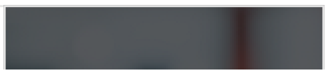
Finally, there a list of recommended readings:

Advances in Financial Machine Learning

Machine learning (ML) is changing virtually every aspect of our lives. Today ML algorithms accomplish tasks that until…

www.amazon.com





Online Lectures in Statistical and Financial Topics

Hands-on Lectures for Quantitative Finance: Teaching Scientific and Statistical Rigor Through Real Data Examples

www.quantopian.com

https://www.amazon.com/Machine-Trading-Deploying-Computer-Algorithms/dp/1119219604

Good luck and stay tuned! The last I want to add — don't give up! I know that this article is kind of discouraging, but actually it doesn't differ that much from the other areas where machine learning is applied. You think healthcare applications where patient life depends on a mathematical model is different? Or marketing applications? Or anything else? Anyway you have to study the business field, model it properly and only after apply ML. The same in finance — we just need to model the market a bit more careful :)

P.S.
Follow me also in Facebook for AI articles that are too short for Medium, Instagram for personal stuff and Linkedin!