Quantinvest Competition Report

Augmenting economic theory on portfolio creation
with machine learning techniques

The Groovy Weasels

Krzysztof Kowalczyk

Mikołaj Ostrzołek

Faculty Advisor

Dr Marcin Chlebus

Abstract

In this report, we evaluate various approaches of applying advanced analytics methods, including machine learning, to portfolio construction. In particular, we analyze ways in which application of deep learning can augment classical economic theory on portfolio creation, in order to create more effective strategies without violating underlying assumptions about economic markets. We establish that using recurrent neural network as a covariance forecasting tool, can provide data more relevant in terms of portfolio construction than traditional covariance shrinkage approach. We then demonstrate a specific portfolio creation strategy that, on average, outperforms classical strategy on the historical data provided for the competition.

The first section of this document explains foundations of the theory upon which our research was based, and explains classical portfolio creation methodology. In the second section, we briefly describe our experiments from the first stage of the competition and their immediate implications, which led to a final experiment described in the third section. Our key findings' source code is available in the summary notebook[1] along with visualization notebook[2]. We have also published rest of our codebase[3], including first-stage experiments, which were left untouched as there was no need to reproduce them to reproduce the main portfolio creation process that we have settled on. Appendix A contains results of initial experiments, Appendix B is the source code repository containing all our experiments and Appendix C contains visualization of final results, in particular the comparison of augmented and classic portfolio building strategies.

Introduction

Theoretical background

Markowitz

In 1952 Harry Markowitz published a paper giving foundation to the 'Modern Portfolio Theory'. It enabled scientist and investors to create optimal portfolios using statistical tools. Binding returns and risk he has shown the trade-off, but also that diversification can lower the risk without lowering the portfolio return. Having a narrow scope of assets (7) available, which are also strongly connected to the market cycle, limits gains from the diversification.

Efficient-Market Hypothesis

The notion of the unpredictability of the stock market has its foundations in Eugene Fama's paper (1965) and further reviewed and confirmed by for example Burton Malkiel (2003). Our attempts to forecast future returns or variability for a long interval of one year will be then somehow contradicting this theory. Moreover, as we have less than twenty years in our

[1] Summary Notebook
[2] Visualization Notebook
[3] Source code repository

sample and just one full business cycle it will be unlikely to find statistically significant models with a horizon of one year.

Competition data analysis

Distribution

Returns of bond and stock funds are quasi-normally distributed with fat tails what is common for financial time series. The distribution of the cash fund is positively skewed as negative interest rates are uncommon.

Correlation

Simple correlation matrix shows high positive correlation between stock funds, negative correlation between the developed markets bond fund and stock funds, developing markets bond fund is strongly positively correlated with stock funds. Rest of correlations is close to zero. As our data sample is fat tailed we will use Ledoit-Wolf Shrinkage (2003) to avoid estimation error. After transformation we see near-zero correlation.

Baseline approach

In order to conduct meaningful experiments, we needed to create a portfolio strategy based on classical methods, and optimal in terms of competition objective. To do this, we have analysed performance of portfolios optimized in respect to:

a) maximum Sharpe ratio,
b) minimum volatility,
c) target risk (3%, 4%, 5%, 6%, 7%, 8%, 9%)
d) target returns(3%, 4%, 5%, 6%, 7%, 8%, 9%)

We came to the conclusion that most optimal method for client with SRRI 3-4 is choosing the portfolio based on target returns equal to 4% and Efficient Frontier Optimisation using Ledoit-Wolf Shrinkage. Optimising portfolio with higher target returns leads to portfolios with risk exceeding SRRI 4, while optimising using method a), b) or c) results in allocation mainly in cash fund what is not optimal for the client as the expected returns are too low.

Machine learning experiments

This section describes our efforts to check which ways of augmenting the classical portfolio creation process are the most promising, which were performed during the first competition stage. These experiments are only briefly described so that we can focus on the final experiment and its outcome in the next chapter.

Generating optimal portfolio

Our first machine learning experiment was to implement an end-to-end neural network that attempts to build an optimal portfolio. In our case, optimal portfolio allocation has the highest

possible returns during 1 year following its construction, with volatility kept below 10%, as required for SRRI level 4[4].

Our neural network was implemented in PyTorch[5] framework and consisted of LSTM[6] and fully connected layers, with softmax function at the end. Input data consisted of the time series of the 7 funds, scaled to zero mean and unit standard deviation. The model was trained in a supervised way, with labels for each data point being the optimal portfolio allocation for the following year (following 260 data points). These labels were not calculated explicitly, but instead we relied on a custom loss function, that penalized excessive volatility and low returns. We have experimented with two different loss functions:

$$loss_1 = (\sigma_p - \sigma_t)^2 - r_p \cdot w_r$$
$$loss_2 = (\sigma_p - \sigma_t)^2 - exp(-r_p) \cdot w_r$$

where: σ_p - generated portfolio volatility, σ_t - targeted volatility - hyperparameter
r_p - generated portfolio returns, w_r - weight of return penalty - hyperparameter

Unfortunately, the model did not perform well during a validation period of the final 2 years in the dataset[7]. Our initial conclusion was, that the period of 1 year ahead, that we needed to forecast, was long enough to introduce major shifts in economic market trends, which the model simply could not take into account given the data that it had. Looking back, it was also possible that the model was not able to build a relevant internal representation of the data without being trained for an auxiliary task such as forecasting returns or fund covariance. Such technique is quite common in natural language processing and reinforcement learning, and checking whether it might work in case of portfolio creation is one of the further research topics we'd like to suggest.

Covariance forecasting

Forecasting future covariance of fund values based on their historical behaviour and additional market data could be a better input to the classical portfolio-building methods than a standard Ledoit-Wolf covariance. To test whether this approach is viable, we have trained a LightGBM[8] regression model, predicting each value in the future covariance matrix individually. As an input, it received historical covariance of fund returns, calculated in daily, monthly and yearly time windows, along with time series features calculated on additional data sources (index values from all around the world and currency exchange rates).

---

[4]Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam (2017). Automatic differentiation in PyTorch,

[5]Synthetic Risk and Reward Indicator, Committee Of European Securities Regulators, Date: December 2009, Ref.: CESR/09-1026

[6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[7] Source code: LSTM (1st attempt)

[8] Ke, Guolin and Meng, Qi and Finley, Thomas and Wang, Taifeng and Chen, Wei and Ma, Weidong and Ye, Qiwei and Liu, Tie-Yan (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems 30 (NIPS 2017).*

The model did not manage to predict covariance values with desired accuracy, which means either we did not have the right additional data to capture long-term changes in relative behaviour between funds, or that there is no way to forecast covariance on such a long period.[9] It was also possible, that LightGBM with its single-value-only regression cannot be trained to take relationships between target variables into account. We left these two hypotheses for further research during the second stage of competition, as we wanted to try out one last idea before the first stage ended.

Model Ensembling

Ensembling many models yields better results than any of the single estimators[10] for a variety of problems, and naturally it seemed like a valuable experiment to conduct. To do this, we have generated portfolios[11] for 18 strategies using PyPortfolioOpt[12] package, which allows for easy application of portfolio creation theory. All of the base portfolios use Ledoit-Wolf Shrinkage and historical returns as their input.

LightGBM Ensembling

At first, we decided to train a simple LightGBM model for a classification problem - predicting which of the base models will have the highest returns in a following year. As an input, the ensembling (2nd-layer) model received allocation of each base portfolio along with metrics: expected annualized returns, expected volatility, sharpe ratio. The model was trained on the initial 64% of a dataset (sorted chronologically to prevent information leaks from the test set), and validated on a following 16% of a dataset for early stopping to prevent overfitting. The remaining 20% of the dataset was used to test the final model.

The test set accuracy of the model is initially quite high (40-50%), but drops quickly[13] as the test data starts following different trends than training data. Given that selecting the base estimator that achieved the best results most often in the history is expected to be 30% accurate, this approach quickly becomes impractical. The results are nonetheless promising and made us explore more sophisticated ensembling techniques, which use an neural networks to explicitly optimize for the selected portfolio performance (the classification of which model is the best one did not carry information about expected performance of the chosen portfolio)[14].

LSTM ensembling

The idea is simple: train a neural network that outputs portfolio allocations for the funds, exactly as in the “Generating optimal portfolio” subsection, with exception that we also feed the base estimator portfolios as an input to the network. Failing to use the target and loss

[9] Source code: Covariance Forecasting
[10] Hendrik Jacob van Veen, Le Nguyen The Dat, Armando Segnini (2015). Kaggle Ensembling Guide. *https://mlwave.com/kaggle-ensembling-guide/*
[11] Source code: Stacking portfolio generation
[12] PyPortfolioOpt. *https://pyportfolioopt.readthedocs.io/*
[13] See Appendix A
[14] Source code: LightGBM Ensembling

function described in previous subsection, we used a loss function that penalizes low returns less if the base models could not achieve high returns as well:

$$loss = exp(r_t - r_p)$$

where: r_p - returns from the generated portfolio, r_t - from the best-performing base portfolio

Unfortunately, we were unable to train the network to achieve a reliable test set performance. We have experimented with 3 different LSTM-based architectures, as well as with a simple fully-connected network, all with little success. Plotting loss function value over the validation[15] period reveals that for some days sub-optimal choice of portfolio can lead to a much worse performance than the optimal one, causing large spikes in loss function that none of the models could optimize for[16].

First stage conclusions

The instability of ensembled models' predictions made it even more plausible that we didn't have a way to capture macroeconomic shifts driving the market changes in a period as long as 1 year. We also established that training a model with an architecture that either does not take relationship between target variables into account, or is very complex is not a viable solution, and constructing an end-to-end machine learning portfolio creation tool is impossible without finding the appropriate set of features and architecture that can successfully predict economic market trends based on them.

Because of these problems, after the first stage we have decided to settle on one of the traditional portfolio creation techniques, which appeared the most reliable historically. Choosing one of the base-level portfolio generators allows us for the complete interpretability of our model, which may be necessary, especially in the scenario in which both ensembled and base-level portfolios perform poorly.

Advanced forecasting

Knowing we lacked features capturing changes in economic trends over a year-long period, in the second stage of competition we decided to focus on finding features that would allow a relatively simple model predict future covariance and fund returns. If we managed to train such model, it could serve as an input for the efficient frontier portfolio creation strategy. If not, if would reassure our belief that portfolio allocation should be performed based on fundamentals of portfolio theory and economics, rather than based on advanced analytics and forecasting.

Dataset construction

External data

In order to tackle the problem of highly correlated features, we decided to manually select which funds and indices are included in the training data. We chose to use at most one index

[15] See Appendix A
[16] Source code: LSTM Ensembling

or indicator per asset class related to the 7 funds between which we have to allocate our portfolio. For this purpose, we selected:

- CRY - Thomson Reuters/CoreCommodity CRB Commodity Index, which measures the average price of key commodities
- SPX - S&P 500® index, that includes 500 largest publicly traded companies in the USA
- FX.F - Euro Stoxx 50 index, that includes 50 largest public companies in the Eurozone
- WIBOR 3M rate for PLN, which reflects the state of economic cycle in Poland
- LIBOR 3M rate for EUR and for USD, which reflect state of developed economies
- Dollar Index, which depends on exchange rates between US dollar and other currencies
- WIG20 index, which includes 20 largest Polish public companies

Daily open, high, low and close values of these indicators were downloaded from Stooq and are publicly available. Some indicators also included trading volume information. All of these values were then put through a feature generation process described below.

Feature generation

In order to easier capture patterns in our data we have decided to include additional features. Following Bao, Yue and Rao (2017)[17] we take advantage of technical analysis indicators. We do not assume a priori any particular relationship between them and future changes in price/volatility. However, we suppose they may be useful in representing the current state of the market.

- Bollinger Bands (120-day intevals), useful indicators of market stability, the closer they are, the less volatile the market is. Moreover, they also help to assess if the price is relatively high or low.
- Exponential moving average (240-day interval)
- Moving Average Convergence Divergence (MACD) by combining sensitive (80-day interval) and more stable (160-day interval) exponential moving average, it can capture changes in the trend of the asset
- Stochastic Momentum Indicator[18] (160-day inteval)
- Money Flow Index (120-day intrval)

[17] *Bao, W., Yue, J., Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory*

[18] SMI description

All of the feature were calculated using Technical Analysis Python library[19]. For series that had open-high-low-close values or the volume we could generate more features than for fund values themselves, therefore better capturing market mechanisms. Entire feature generation process is defined in the summary notebook[20].

Data preparation for training

Generated features were combined together with past values of target variables, and processed into a sequences, so that they can be read by a recurrent neural network such as our LSTM described in a following subsection.

We intended the models to forecast one-year-ahead values of each fund as well as funds' values covariance matrix during the following year. In order to remove noise from target variables, we used shrunk covariance matrices (using Ledoit-Wolf Shrinkage, just as in our base model).

In the end, we had 134 features and 56 targets for each sample (series).

Model architecture and training

The architecture we decided to settle on was a most basic version of an LSTM multivariable regression model, once again implemented in PyTorch framework. The model consisted of a double LSTM layer and one fully-connected layer. Hidden dimension size of LSTM layers was set to 80 and input sequence length to 520 days (2 years). That way the input was long enough, but the number of samples in a training window of 10 years was still large enough.

The model was trained for 10 epochs, with a batch size of 1 series, and an Adam optimizer (Kingma & Ba, 2014) set to 0.0002 learning rate, optimizing mean squared error loss.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Some hyperparameters (hidden dimension size in LSTM, sequence length and learning rate) were optimized through a grid-search.

Portfolio creation

For a given model prediction, which consisted of 7 expected future fund returns values as well as 49 variables representing unraveled fields in a covariance matrix, we needed to generate a portfolio allocation between the 7 funds. We evaluated 4 portfolio creation strategies:

- "BASE" - baseline strategy, which disregards model predictions and uses mean historical returns and Ledoit-Wolf Shrinkage
- "LSTM" - uses returns and covariance forecasts from LSTM output, disregarding classical methodology

[19] Technical Analysis Library
[20] Summary notebook

- "LSTM-COV" - hybrid strategy, uses covariance forecast from LSTM, but relies on classic methodology for expected returns (mean historical returns), breaking assumption that the financial instrument covariance is constant in time
- "LSTM-RET" - hybrid strategy, uses returns forecast from LSTM, but relies on Ledoit-Wolf Shrinkage calculated on past covariance, assuming covariance is constant in time but breaking efficient market assumption

The strategies defined above specify inputs to efficient frontier optimization, which in all 4 cases targets "efficient risk" - maximizing returns with constraint that expected volatility has to be kept below 10% (a maximum for SRRI level 4).

Performance evaluation

The model needed to be trained multiple times on different training and validation intervals to make sure its behaviour is stable and performance reproducible. In order not only to analyze the model itself, but the portfolios that can be created based on its data, we have implemented a validation loop consisting of 6 iterations. Each iteration consisted of:

1. Selecting a 10-year training interval, immediately followed by 1-year test interval
2. Fitting a model on 90% of the training interval and calculating validation loss on the remaining 10% of the training interval to prevent overfitting
3. Generating model predictions for the period of test interval, and constructing a portfolio based on these predictions using one of the strategies described in the preceding subsection
4. Calculating portfolio returns and volatility for the test period

Because of the need to calculate features as well as target variables, all of which took form of sequences, the performance evaluation was performed 6 times. For details, see summary notebook[21] which contains implementation and results of this process. The visualization and comparison of models' results are also included in Appendix C to this document.

Final strategy selection

Choice of the final strategy must be optimized under two dimensions: returns (as high as possible) and risk (SRRI between 3 and 4 what corresponds to the volatility between 2% and 10%). Comparing 4 methods: *BASE, LSTM, LSTM-COV* and *LSTM-RET* in the dimension of returns, LSTM clearly outperforms other (Figure 5). However, on the Figure 6 we can see that it is at the expense of the higher volatility. Nevertheless, it is still always bounded by the 7%, thus does not exceed the SRRI 4. We recommend using the strategy based on the LSTM to achieve the best results when client is still far from the retirement age. However if the appetite for the risk of customer will fall when he will be closer to the retirement age we recommend switching to less volatile strategies which combine LSTM with more traditional approach. In particular, LSTM-COV seems very appealing with low volatility, returns slightly higher than the BASE, and without breaking assumption of market efficiency from the BASE strategy.

[21] Summary notebook

Bibliography:

Bao, W., Yue, J., Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory

Fama, E. (1965). Random Walks in Stock Market Prices. *Financial Analysts Journal, 21*(5), 55-59. Retrieved from http://www.jstor.org/stable/4469865

Hendrik Jacob van Veen, Le Nguyen The Dat, Armando Segnini (2015). Kaggle Ensembling Guide. *https://mlwave.com/kaggle-ensembling-guide/*

Ke, Guolin and Meng, Qi and Finley, Thomas and Wang, Taifeng and Chen, Wei and Ma, Weidong and Ye, Qiwei and Liu, Tie-Yan (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems 30 (NIPS 2017).*

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Ledoit, O., & Wolf, M. (2003). Honey, I shrunk the sample covariance matrix. *UPF economics and business working paper*, (691).

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of economic perspectives*, *17*(1), 59-82.

Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance,7*(1), 77-91. doi:10.2307/2975974

Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam (2017). Automatic differentiation in PyTorch,

Appendix A: Selected results visualization

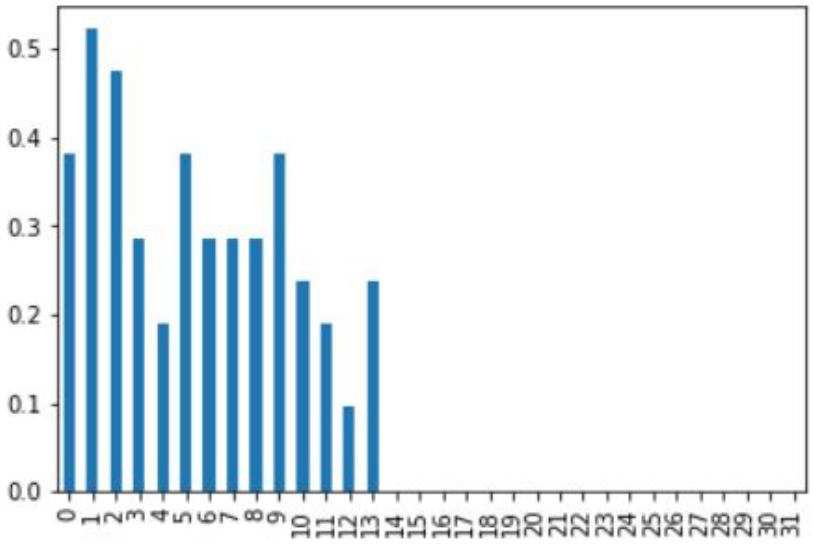

Figure 1. Model accuracy on the test set, X axis represents index of a month in the test set,Y axis represents average accuracy of the model prediction in that month

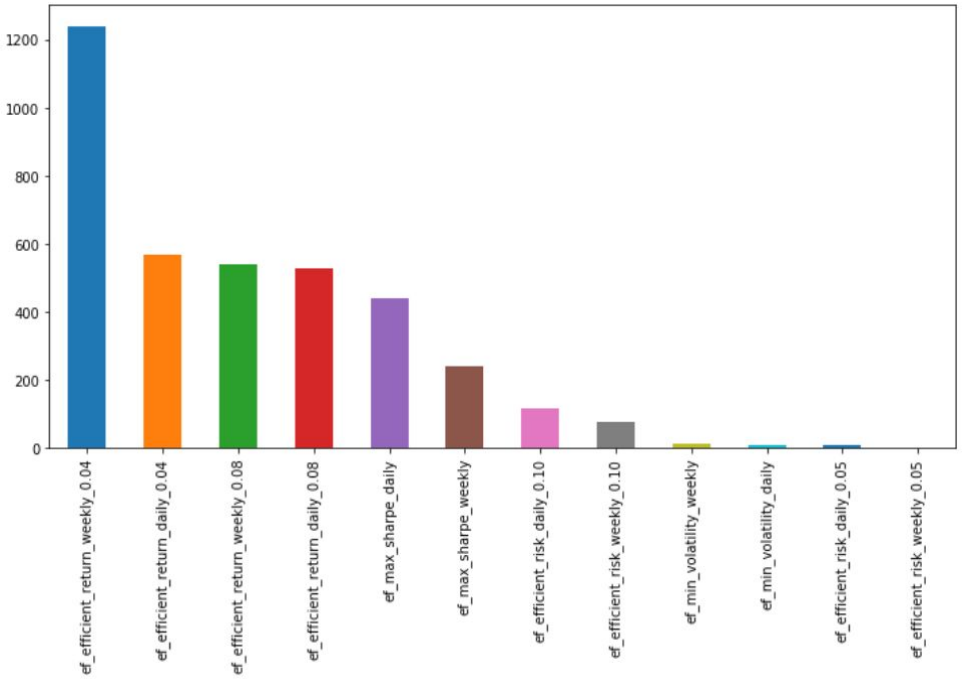

Figure 2. The base-level portfolio that targets 4% annualized return and minimizes volatility turns out to be the best in over 30% of time intervals.

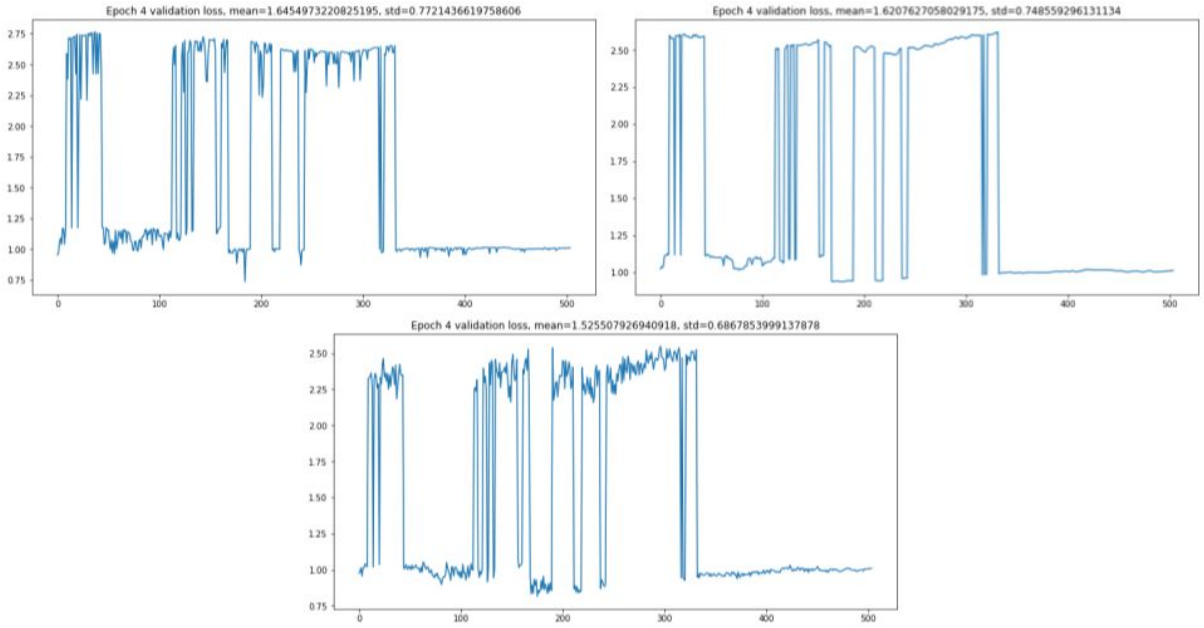

Figure 3. Validation loss of 3 different 2nd-level models - only on some occasion the ensembled model performs better than all of the base-level portfolios (loss values below 1.0). Choosing sub-optimal weights (loss spikes) happens nearly as often as choosing optimal ones.

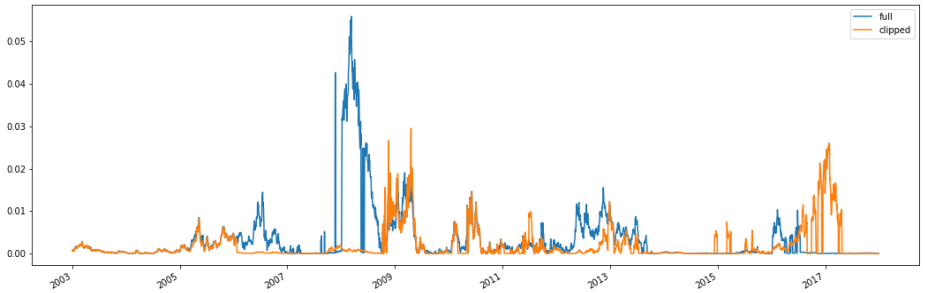

Figure 4. Variance of 18 base portfolios' returns (on a year following its construction).
Full - portfolios constructed from all available historical data,
Clipped - portfolios constructed from at most 5 years of historical data.

Appendix B: Experiments source code

All of the source code is available at: https://github.com/kowaalczyk/quant-invest

Appendix C: Final results

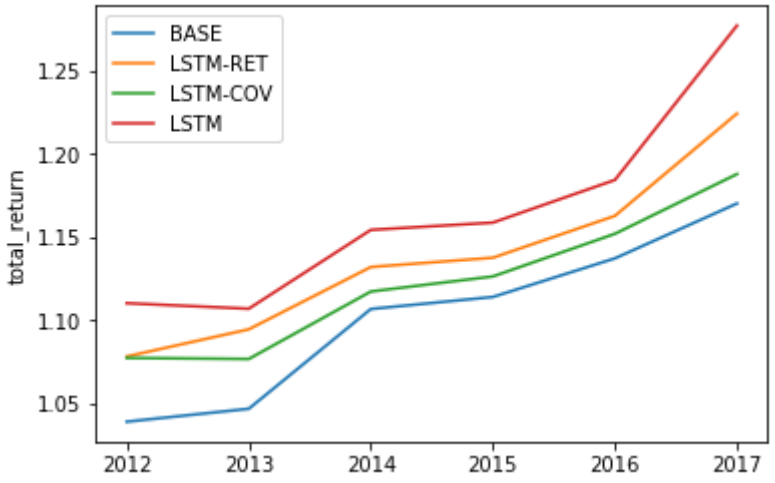

Figure 5. This graph clearly shows that all all methods tend to be connected by the common market trend.Simple LSTM clearly outperforms other approaches in terms of returns.

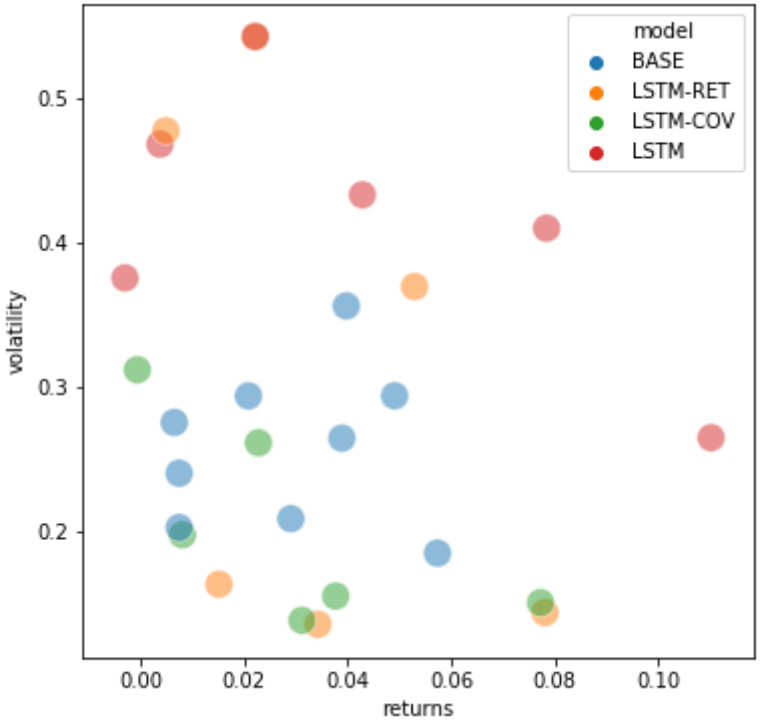

Figure 6. Comparison of 4 methods on 6 one-year periods (2012,…, 2017). We can see that simple LSTM is characterised by high volatility, baseline method has low volatility but achieves the lowest returns on average. LSTM-RET and LSTM-COV appear somewhere in the middle.

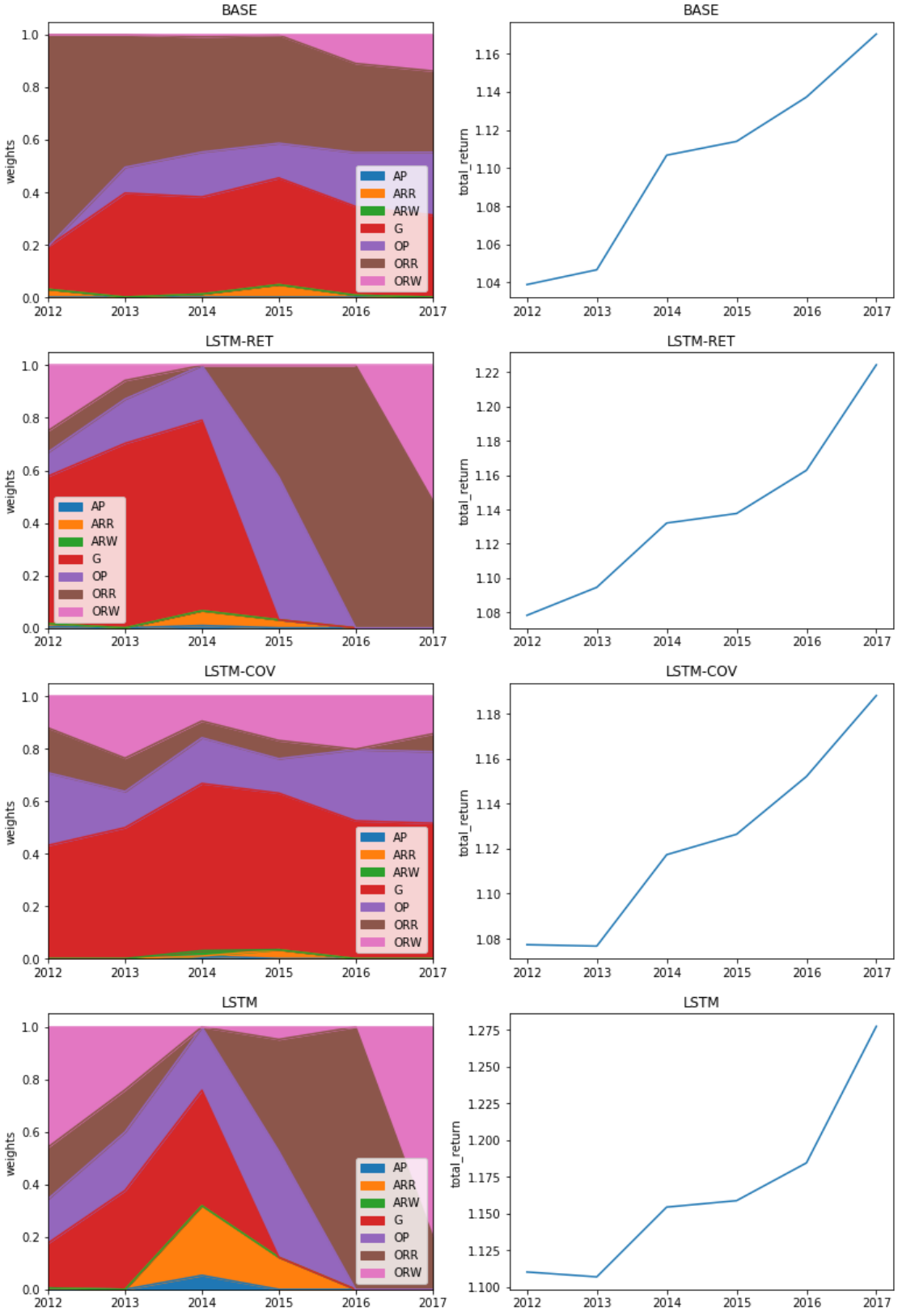

Figure 7. Weights and returns