COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

IEORE4720 Deep Learning– Final Project

Deep Learning for Finance – Deep Portfolio

April 2018

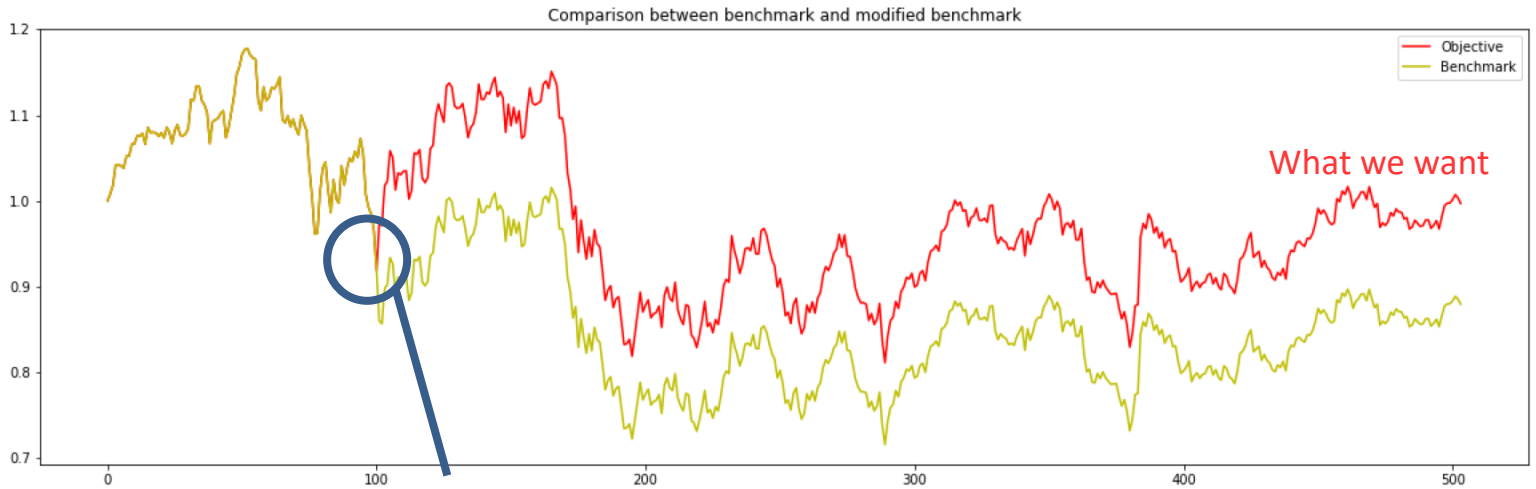
Group 5 - Members

Thibault Becker
Marin Bergerot
Paul De Kerdrel
Achille Mascia
Nicolas Tachet

Introduction

Objectives:

1. Reproduce an index (IBB) by finding the right selection of investments using Deep Learning techniques
2. Reproduce an index (IBB) with anti-correlation in periods of large drawdowns to outperform the index i.e. reproduce a modified benchmark
3. Extend this technique to other index (S&P 500)

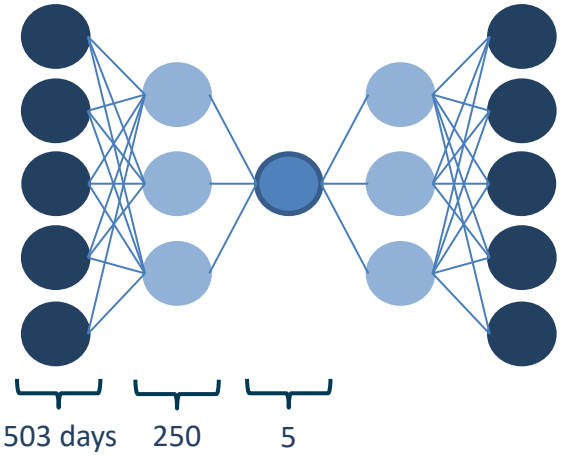


What we call a "*large drawdown*"

Step 1: Autoencoder

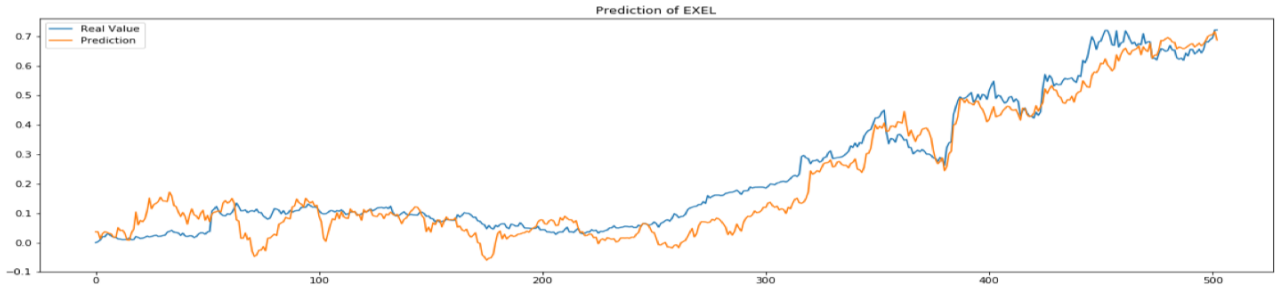
Daily rate-of-returns of each stocks in the index over 503 days

INPUT

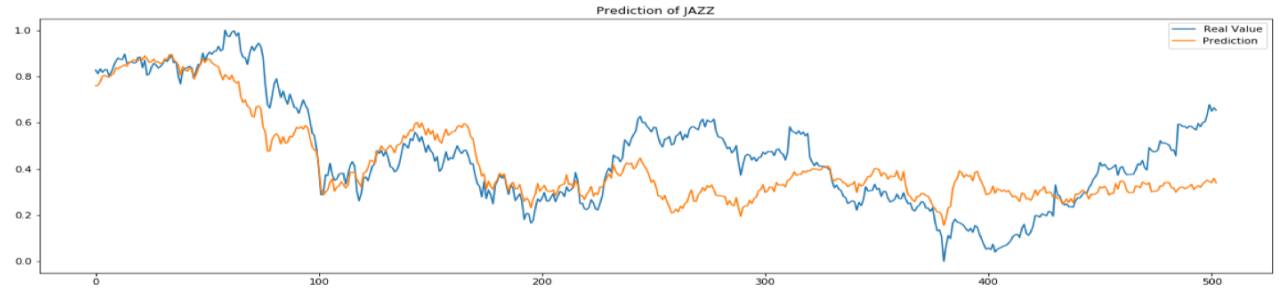


1. **Reproduce** each stock
2. **Identify** the most/least communals

OUTPUT



⇒ Best communal



⇒ Least communal

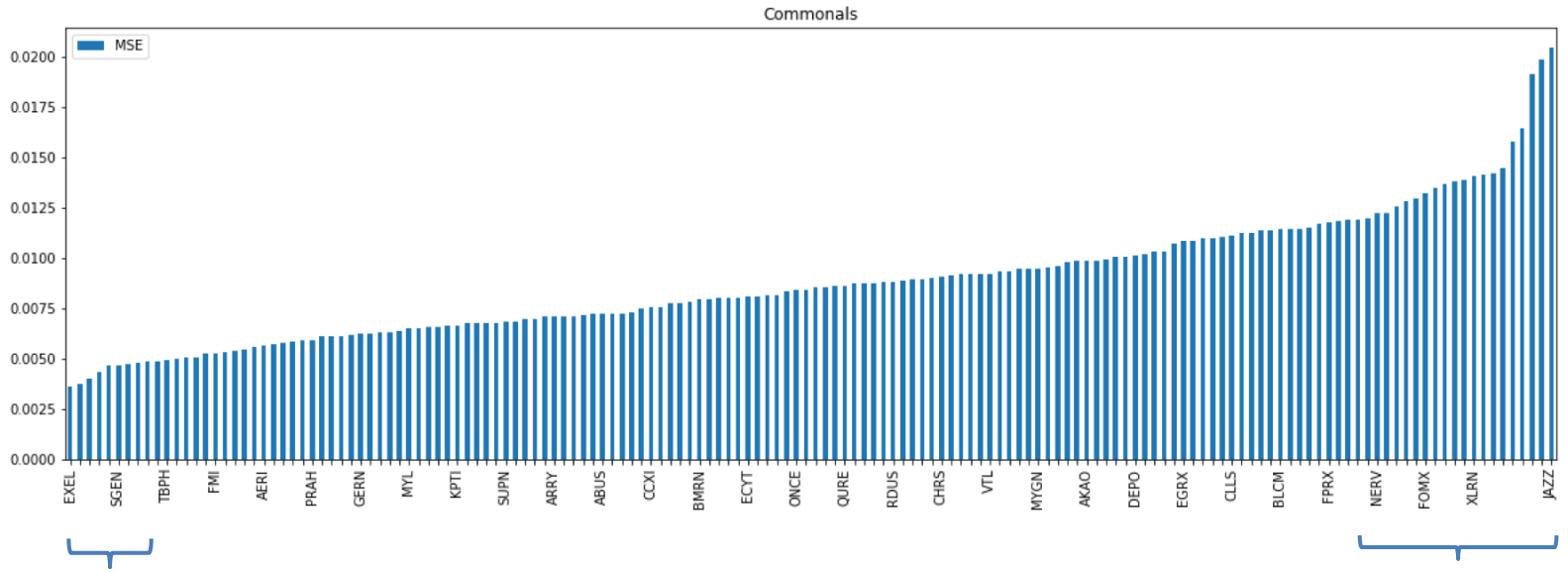
Step 1: Autoencoder

Parameters:

- learning_rate = 0.0004
- num_steps = 5000
- batch_size = 30
- lambda = 0.12
- val_dropout = 0.89

Choose our stocks:

The proximity of a stock to its auto-encoded version provides a measure for the similarity of a stock with the stock universe.



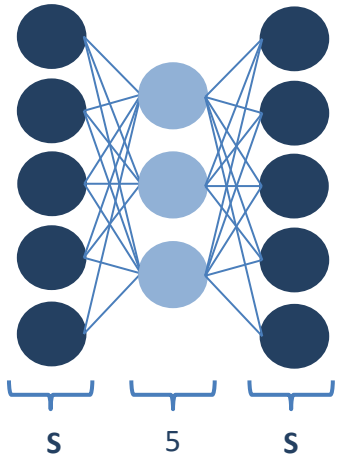
10 most communals

S - 10 least communals

Step 2: Calibration

S **Stocks** (log rate-of-returns)

INPUT

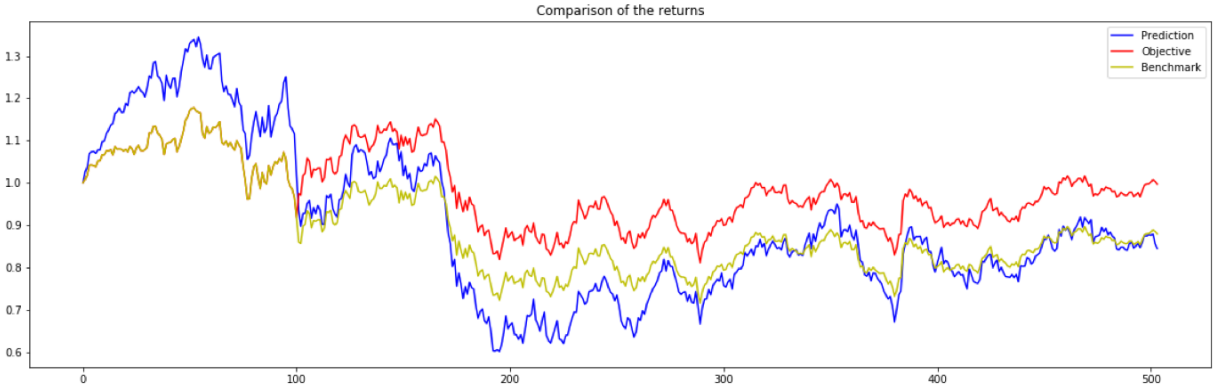


Weights of the portfolio that reproduces the index

OUTPUT

Parameters:

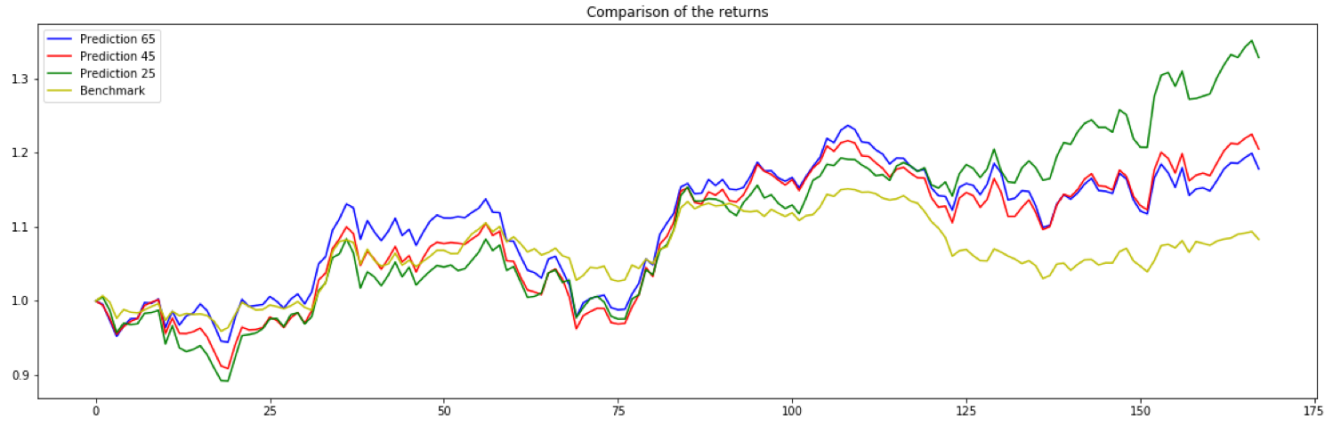
- learning_rate = 0.0004
- num_steps = 1000
- batch_size = 10
- lambda = 0.1
- val_dropout = 0.88



Step 3: Validation (or Out-of-sample phase)

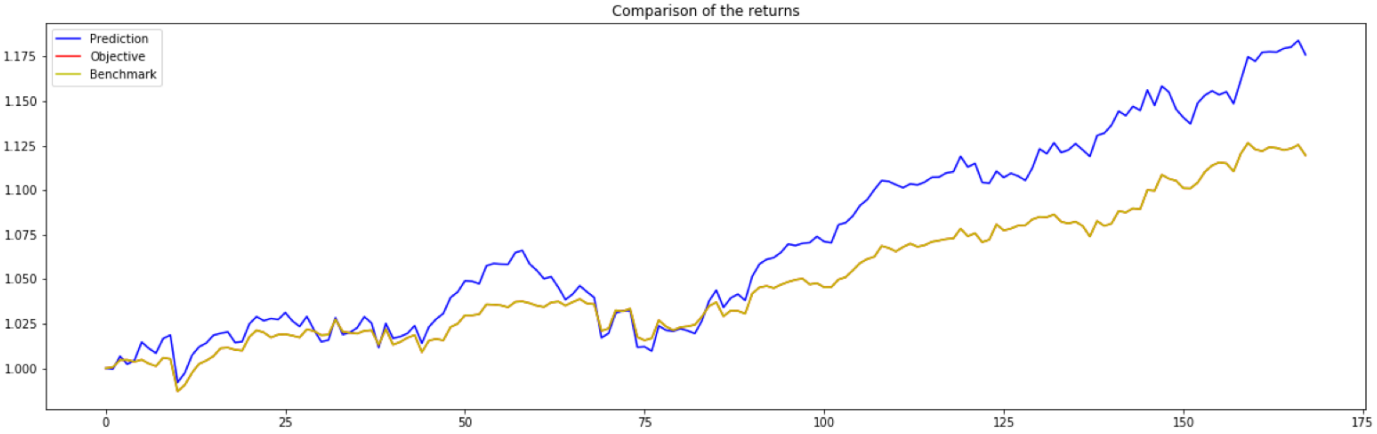
Results for S = 65

Results for different S values



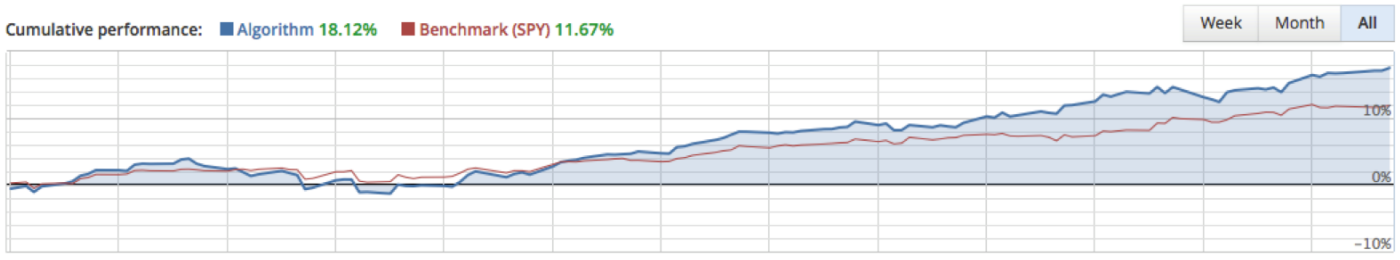
Extension with the S&P

Results out-of-sample:



Analyze our strategy

Total Returns	Benchmark Returns	Alpha	Beta	Sharpe	Sortino	Volatility	Max Drawdown
17.53%	11.91%	0.05	1.23	3.68	5.79	0.09	-5.04%



Conclusion

Does Deep Learning make a difference?

Machine Learning Pipeline

S&P 500 Clustering: using k-medoids algorithm

⟺ *Autoencoder*

Replicate S&P: Construct mini-portfolio

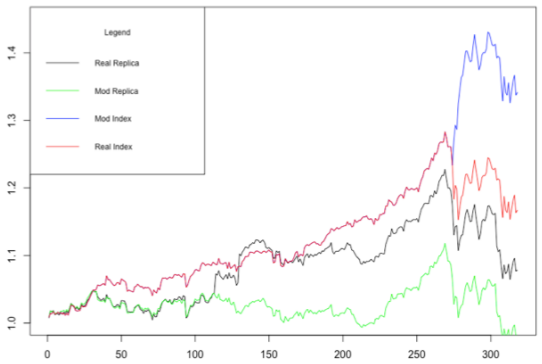
⟺ Choose S commonals

Weights allocation

$$\min \left\| Y_{ind} - \sum_{i=1}^{k} \omega_i Y_i \right\|_2^2 \quad s.t. \quad \sum_{i=1}^{k} \omega_i = 1$$

⟺ Calibratrion (NeuralNet)

Results with ML



- Real index mean: 0.000266
- Portfolio mean: -3.562e-05 (Severely underperform)

Conclusion

By uncovering deep features using our Neural Networks, we observe a performance improvement compare to ML techniques. We are able to replicate and beat our benchmark.