# Roboadvisors, quant algorithms inside the robots

## Omnibus Presentation

Emiel Lemahieu

InvestSuite, Ghent University

*emiel.lemahieu@investsuite.com*
*emiellemahieu.github.io*

March 8, 2023

# Contents I

1 Introduction
- Roboadvisors, ¿Qué
- InvestSuite
- Quantitative methods inside the robots

2 Levelling up with AI
- AI, ¿Qué
- AI in portfolio construction
- Application: Market Generator
- Application: Model Interpretability

3 Robo trading demo
- Introduction to the skateboard approach
- Demo
- Extensions and challenges

4 Conclusion

# Disclaimer

*Views and opinions expressed are my own and do not necessarily reflect the views and opinions of InvestSuite. I am not a financial advisor. None of the information should be taken as financial advice or recommendations. All my writings are for informational purposes only. All provided codes are to be used at own responsibility.*

# Roboadvisors: ¿Qué

**ROBO-ADVISORY: FROM INVESTING PRINCIPLES AND ALGORITHMS TO FUTURE DEVELOPMENTS**

Chapter to appear in the book
*"Machine Learning in Financial Markets: A Guide to Contemporary Practice"*
Cambridge University Press, 2021. Edited by A. Capponi & C. A. Lehalle

ADAM GREALISH[a] AND PETTER N. KOLM[b]

ABSTRACT. Advances in financial technology have led to the development of easy-to-use online platforms referred to as *robo-advisors* or *digital-advisors*, offering automated investment and portfolio management services to retail investors. By leveraging algorithms embodying well-established investment principles and the availability of exchange traded funds (ETFs) and liquid securities in different asset classes, robo-advisors automatically manage client portfolios that deliver similar or better investment performance at a lower cost as compared to traditional financial retail services.

- **Automated investment advice**
- Many savers turned into investors (e.g. digital natives)
- Paper 'Robo-Advisors Today and Tomorrow: Investment Advice Is Just an App Away.'

InvestSuite

**Key Facts**

- B2B roboadvisor based in Leuven
- Founded in 2018
- Since then grew to a team of 70+ people
- 25 clients on 3 continents
- 5 products in our current suite (including Portfolio Optimizer)
- www.investsuite.com

# Quantitative methods inside the robots


- **Portfolio construction in practice**
- Large gap between academia and existing robo solutions
- Beketov (2018) investigated 219 robo-advisors covering the vast majority of market players (including Wealthfront, Betterment, etc.)
- Link to article

# Quantitative methods inside the robots

**Table 3** Occurrence of different methodological frameworks within the Robo Advisors analyzed

| Methodological framework | Occurrence (%) |
|---|---|
| Modern Portfolio Theory | 39.7 |
| Sample Portfolios | 27.4 |
| Constant Portfolio Weights | 13.7 |
| Factor Investing | 2.7 |
| Liability-Driven Investing | 2.7 |
| Risk Parity | 1.4 |
| Full-Scale Optimization | 1.4 |
| Constant Proportion Portfolio Insurance | 1.4 |
| Mean Reversion Trading | 1.4 |
| Other | 8.2 |

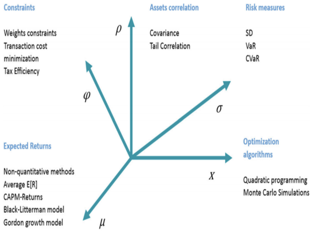
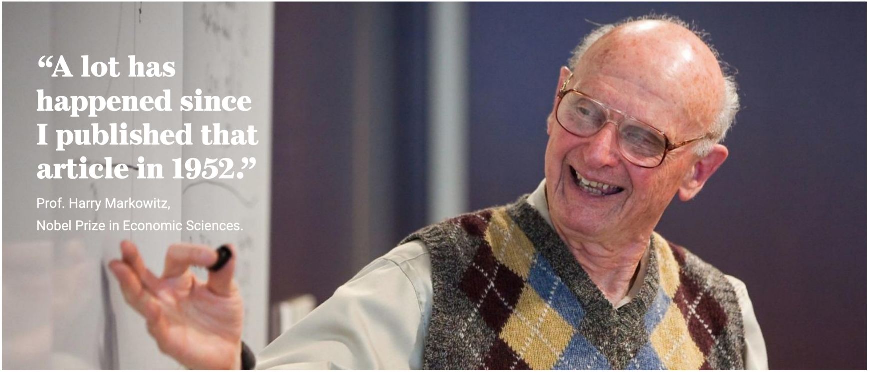



**Fig. 3** Schematic of the "Multidimensional improvement of Modern Portfolio Theory" in Robo Advisors. The methods mentioned are those that used in RAs, and they do not comprise a comprehensive list of the methods that are or can be used to improve the Modern Portfolio Theory framework in general. Note: VaR and CVaR optimization are frequently considered to be alternatives to Modern Portfolio Theory rather than improvements to this framework

# Quantitative methods inside the robots



# Quantitative methods inside the robots

A lot has happened indeed...

Should I still use Markowitz mean-variance optimization now that AI has arrived?

Markowitz mean-variance optimization is still a widely used method in finance, especially in portfolio optimization. The arrival of AI has led to the development of more sophisticated techniques, but this doesn't necessarily mean that the mean-variance optimization is no longer relevant.

In fact, some modern AI techniques such as reinforcement learning and deep learning are built upon the foundation of mean-variance optimization and other classical financial models. AI can also enhance the mean-variance optimization by providing more accurate estimates of expected returns and covariances, or by incorporating additional information and constraints.

In summary, Markowitz mean-variance optimization remains a valuable tool in finance and can be further improved with the help of AI.

Artificial intelligence, what’s in a name?

# Levelling up with AI

- AI: automation of tasks that require cognition (e.g. portfolio construction)
- ML: learning from data (decision rules are output not input)

*Disclaimer: it's all just fancy statistics really*.

# Levelling up with AI: learn from data


Figure: Model driven = we impose our view of how the data behaves in the form of rules.

# Levelling up with AI: learn from data

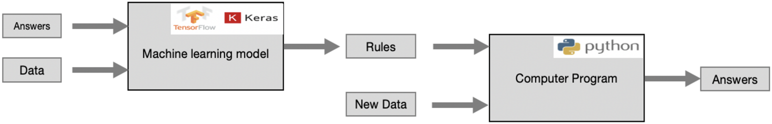



Figure: Data driven = the machine learns the data's view with rules as model output rather than model input.

# Welcome to the zoo

**Welcome to a zoo of more or less wild models to be tamed**

- Generalized / penalized linear models
  - LASSO, ridge, Elnet, PCA regression, PLS,...
- Tree-based methods
  - Decision trees, random forests, ensemble methods (bagging, boosting),...
- Graph-based methods
  - Graph theory (MSTs)
  - Hierarchical clustering methods
- Neural networks
  - Simple ANN or Feedforward NN (MLP-Multilayer perceptron)
  - Convolutional neural networks (CNN, TCN)
  - Recurrent (gated) networks (GRU, LSTM, WaveNet)
  - Graph nets (GNN, GraphSage)
  - ...

# Levelling up with AI: the Bible

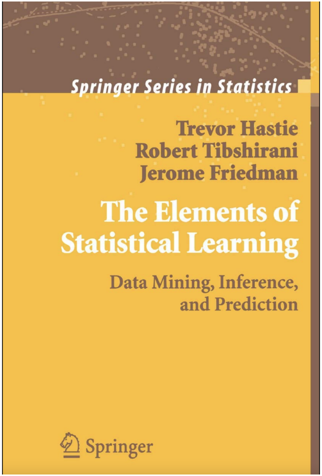



Figure: The Bible (link)

**FAQ1: Where to start with ML in Finance?**

A roadmap (just a recommendation):

1. Start off with Google Machine Learning Crash Course (ML Concepts, Tensorflow, Colab): `https://developers.google.com/machine-learning/crash-course`
2. Skim the ESL (Bible of ML): Elements of Statistical Learning by Robert Tibshirani et al `https://hastie.su.domains/Papers/ESLII.pdf`
3. Skim Machine Learning in Finance by Mat Dixon `https://link.springer.com/book/10.1007/978-3-030-41068-1`
4. Find an interesting problem that you want to play with, like building your own robo trader (see demo below) and learn Python and ML packages while doing (and have fun but keep on reading)!

# Core idea 1: approximate sparsity

We live in a high-dimensional world, driven by low-dimensional rules!
ML allows to find traces of those rules in the data the world generates.
**Approximate sparsity**

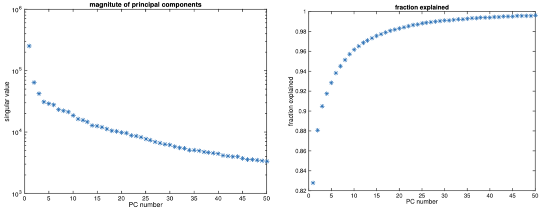



Figure: Approximate sparsity or the sorted absolute values of the model's coefficients would decay fast, i.e. the $j_{th}$ largest coefficient (absolute value), $|\beta|_j \leq A_j^{-a}, a \geq 1/2, \forall j$.

# Core idea 2: universal approximation

*ORIGINAL CONTRIBUTION*

**Approximation Capabilities of Multilayer Feedforward Networks**

KURT HORNIK

Technische Universität Wien, Vienna, Austria

(Received 30 January 1990; revised and accepted 25 October 1990)

**Abstract**—*We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^p(\mu)$ performance criteria, for arbitrary finite input environment measures $\mu$, provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.*

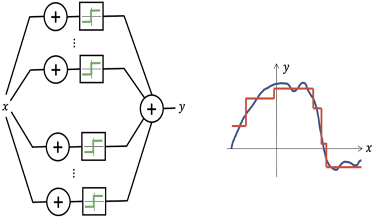



Figure 1: Multilayer Perceptrons (Rosenblatt, 1958), the simplest feed-forward neural networks, are universal approximators: with just one hidden layer, they can represent combinations of step functions, allowing to approximate any continuous function with arbitrary precision.

- Universal Approximation Theorems (e.g. Rosenblatt 1958, Hornik 1991)
- Theoretical underpinnings of the successes of deep learning.

# Core idea 2: universal approximation

- Universal Approximation Theorems often rely on something called the **Stone-Weierstrass theorem**.
- It says that subalgebras are dense in some original space under some conditions (of smoothness and point seperations), and typically boils down to much simpler families of functions (e.g. linear regression) working on a transformed space (e.g. power series) being very good approximators of much more complex functions (e.g. polynomials) on the original space.

# Stone-Weierstrass

- **Stone-Weierstrass universal approximation theorem** is therefore a crucial theorem in proving the universal approximating capabilities of neural networks [Cot90].
- Neural networks concatenate linear weighing functions and non-linear sigmoids or ReLU activation functions, that serve as a learnable[1] basis to linearly approximate (the final fully-connected layer) the relationship between any output Y and input X, i.e. where f(X) = Y is some complex polynomial.

[1] I.e. as a function of the network weights.

# Stone-Weierstrass and neural networks

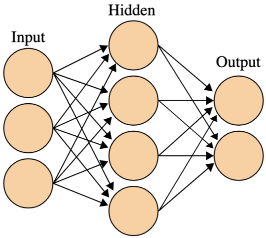



Figure: A simple neural network (from Wikipedia)

# Stone-Weierstrass and neural networks

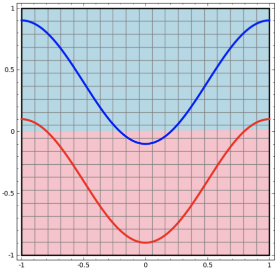



Figure: A simple non-linear problem

# Stone-Weierstrass and neural networks

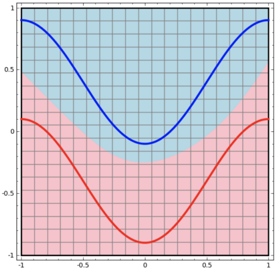



Figure: A simple non-linear solution

# Stone-Weierstrass and neural networks

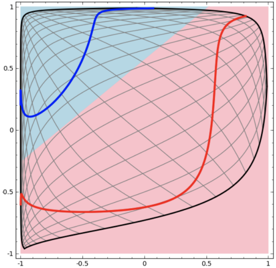



Figure: The transformed hidden or latent space: stretch, squeeze, tilt the space into something useful (a learned representation)!

# Can we use these core ideas for portfolio construction?

Can we use these core ideas for portfolio construction?

# Can we use these core ideas for portfolio construction?

Yes, and no.

# Can we use these core ideas for portfolio construction?

With caution...

# With caution...

INSIDER MARKETS INSIDER Search markets My Watchlist

U.S. MARKETS CLOSED

| DOW 30 | | | S&P 500 | | | NASDAQ 100 | | | GOLD | |
|---|---|---|---|---|---|---|---|---|---|---|
| +1.17% | +387.40 | 33,390.97 | +1.61% | +64.29 | 4,045.64 | +2.04% | +245.94 | 12,290.81 | 0.00% | 1,855.22 |

HOME > NEWS > ETF

## Forget ChatGPT — an AI-driven investment fund powered by IBM's Watson supercomputer is quietly beating the market by nearly 100%

**Phil Rosen** Jan 31, 2023, 2:55 PM

**The Watson-powered ETF is beating a total market fund by nearly 100%.**
PhonlamaiPhoto/Getty Images

**Your Market View**

| NAME / PRICE | +/- | % | DATE |
|---|---|---|---|
| TSLA 197.79 | 6.89 | 3.61% | 03/03/23 9:00 PM |
| AAPL 151.03 | 5.12 | 3.51% | 03/03/23 9:00 PM |
| MSFT 255.29 | 4.18 | 1.66% | 03/03/23 9:00 PM |
| NFLX | 3.30 | 1.06% | 03/03/23 |

Figure: Be critical

# With caution...

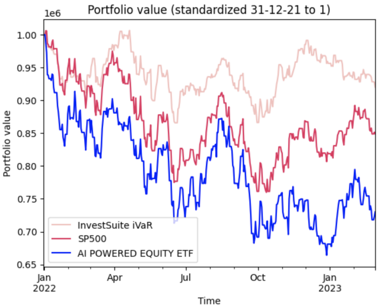



Figure: AI Powered ETF versus benchmark and IVS

# With caution...


DE TIJD

Nieuws Markten LIVE Netto Sabato ABONNEER NU

In Canada gearresteerde topvrouw wordt voorzitter Huawei

Iran 'bereid tot gevangenenruil' met Belgische Vandecasteele

# ‘Robotfonds is als een interessante collega met eigenwijze meningen’

KOPIEER LINK
TWITTER
FACEBOOK
WHATSAPP
LINKEDIN
E-MAIL
BEWAAR
SCHENK DIT ARTIKEL
REAGEER

©Shutterstock

**KRIS VAN HAMME** | 02 september 2021 13:49

Het eerste Belgische beleggingsfonds dat volledig op artificiële intelligentie (AI) draait dwingt zijn menselijke beheerders te graven naar de oorsprong van soms eigenzinnige ideeën. Om na een jaar alles opnieuw heruit te vinden, want winnende formules hebben een kort leven op de beurs.

Figure: Be critical

# With caution...

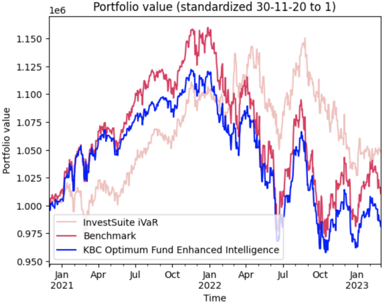



Figure: Optimum Enhanced Intelligence Fund versus IVS

# Can we use these core ideas for portfolio construction?

Financial data (e.g. stock market time series) has its *idiosyncracies*:

- Non-stationarity (statistical relationships like distributions of desired quantities and correlations change all the time because humans change)
- Fat tails (very unlikely scenarios do happen, all the time)
- Low signal-to-noise ratio (many data points are useless, just markets moving 'up and down, sideways and in circles'. Please don't fit your prediction model on Brownian increments).
- ...

# Can we use these core ideas for portfolio construction?

Hence, a more realistic approach is a 3-step data driven optimization approach (abbreviated 3SA):

1. Generate rich scenarios which vary according to some conditions (macro, fundamental, whatever).
2. do optimization.
3. attribute the changes in (1) on the outcomes in (2).

Using ML for (2) is often as dangerous (because of overfitting) as it is tempting, but what will follow focuses on (1) and (3), where I believe it has true value.

# Portfolio drawdown optimization

- We focus on a portfolio drawdown risk ('$\beta$') objective.
- **FAQ2: why not a return ('$\alpha$') objective?**
  Answer: the idiosyncracies of financial data. Using ML as an oracle approach is dangerous and one of the reasons why it is sometimes discredited in the quant community.
- Arguably, the main contribution of quantitative finance is formally managing risk. Risk is easier to predict hence the traditional focus on volatility modeling and risk-based optimization. Moreover, it is much more persistent (e.g. GARCH) rather than (semi-)martingale returns.

# Drawdown

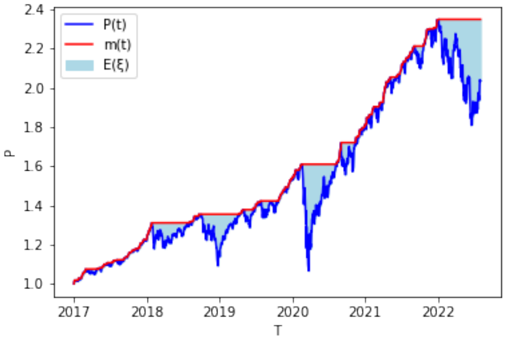



Figure: Example of $\xi$, $m_t$ and $P_t$ for a US Equity index (S&P500)

# Drawdown

For a portfolio value path $S : [0, T] \to \mathbb{R}$, define its drawdown function $\Xi$ as:

$$\xi_t = \Xi(S)_t = \max(\max_{k<t}(S_k) - S_t, 0) \tag{1}$$

From a learning perspective, it is a very ugly thing.

- Differentiability
- Iteration (complexity)

# Linearizing drawdown with path signatures

- Much like a Taylor expansion the drawdown can be linearized with a set of coefficients called the iterated integrals, or *path signature*.

$$\hat{\xi}(N)_t = \xi_0 + \sum_{n=1}^{N} L_n \underbrace{\int_{u_1<\ldots<u_n, u_1,\ldots,u_n \in [0,t]} \ldots \int dS_{u_1} \otimes \ldots \otimes dS_{u_n}}_{\Phi^n(S)} \qquad (2)$$

  where $\hat{\xi}$ is the approximated drawdown at $t$, $L_n$ is a vector of linear coefficients linking the drawdown at $t$ with the signature terms of order n of a path $S$.

- Leveraging factorial decay of the approximation for $Lip(\alpha)$, taking $N \to \infty$ one gets an arbitrarily close approximation of $\xi_t$.

# Stone-Weierstrass, again

- **Stone-Weierstrass universal approximation theorem**: signatures are a universal basis for path functions (with certain conditions of regularity/smoothness). So the proof of Eq. (2) is literally 5 lines of text arguing Stone-Weierstrass is applicable.
- E.g. Martin Hairer's (Field Medalist, 2014) regularity structures uses signatures.

# Drawdown revisited: linear drawdown approximation

Hence, we concluded the paper with specifying:

$$\hat{L} = \min_L (||\Xi(S) - \langle L, \Phi^M(S) \rangle||_2 + \lambda_1 ||L||_1 + \lambda_2 ||L||_2) \tag{3}$$

$$\hat{\Xi}(S) = \langle \hat{L}, \Phi^M(S) \rangle \tag{4}$$

where $\lambda_1 = \lambda_2 = 0$ corresponds to OLS, $\lambda_1 = 0$ corresponds to Ridge and $\lambda_2 = 0$ to LASSO regression [HTFF09].

# Market generator (3SA: step 1)

Can we learn to generate drawdown-realistic market scenarios without assuming a data generating process or drawdown distribution?

# Market generator

- Appreciate that $\hat{\Xi}(S) = \langle \hat{L}, \Phi^M(S) \rangle$ is a differentiable approximation of the non-differentiable $\Xi(S)$.
- Intuitively we just switch between the path space and the signature space to see how a change in a path distribution impacts the drawdown through linear loadings on the transformed basis. We just take a linear combination of a sum, albeit an iterated sum.
- This gives us leeway to embed drawdown evaluation in a (neural) system of differentiable equations, i.e. machine learning!

# Market generator

The general generative architecture that we rely on is a **variational autoencoder (VAE)**. We do not discuss the architecture in depth here, but refer the interested reader to [KW14] for details on encoder/decoder networks $g$, backpropagation, latent Kullback-Leibler $\mathcal{L}_L$ and reconstruction loss $\mathcal{L}_R$.

# Market generator

**Algorithm 1** VAE market generator given important path feature $\Xi$

**Input** Historical price paths $S : [0, T] \to \mathbb{R}^N$, hyperparameters listed above + signature truncation level M and feature weight $\alpha$.

**Output** Trained VAE Market Generator $g_\theta$

**procedure** TRAIN
    Divide historical sample into batches $\mathcal{B}$ of length $\tau$, calculate the signatures of these paths truncated at level M, $\Phi_M^{\mathcal{B}}$, calculate the drawdowns $\Xi$ of these paths $\Xi(S^{\mathcal{B}}) = \int_0^\tau (max_{t_i<t}(S_{t_i}^{\mathcal{B}}) - S^{\mathcal{B}})dt$ denoted $\hat{\Xi}(S_b)$
    $\hat{L} \leftarrow LinearRegression(\hat{\Xi}(S^{\mathcal{B}}), \Phi_M^{\mathcal{B}})$
    Initialize the parameters $\theta$ of the VAE.
    **for** $i : \{1, ..., N\}$ **do**:
        Sample a batch $\mathcal{B}$ and pass it through the encoder $g_\theta$ and decoder network $g_\theta^{-1}$ .
        Calculate drawdown $\Xi(S')$ of the output sample $S'$ using the differentiable signature approximation: $\langle \hat{L}, \Phi_M(S') \rangle$
        Define the reconstruction loss term as the weighted average of RMSE error and . drawdown loss: $\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{\mathcal{B}}||S - S'||^2 + \alpha \mathbb{E}_{\mathcal{B}}||\langle \hat{L}, \Phi_M(S) \rangle - \langle \hat{L}, \Phi_M(S') \rangle||^2$
        $\mathcal{L} = \mathcal{L}_L + \mathcal{L}_R$
        $\theta \leftarrow \theta - l\frac{d\mathcal{L}(\theta)}{d\theta}$

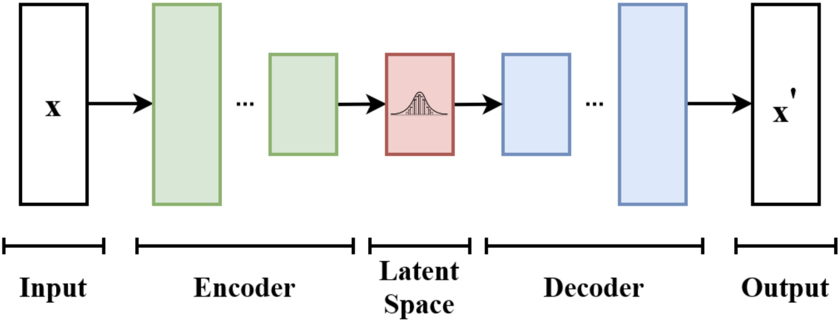
# Market generator

**Algorithm 2** Sampling from the market generator

**Input** Trained VAE Market Generator $g_\theta$.
**Output** $N_s$ Generated samples X'

**procedure** GENERATE
    **for** $j : \{1, ..., N_s\}$ **do**
        Sample a random K-variate Gaussian variable Z
        $X' \leftarrow g_\theta^{-1}(Z)$

# Market generator

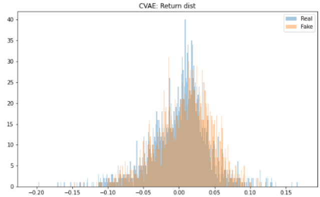



A standard VAE (from Wikipedia)

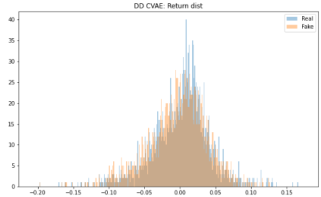
# Market generator

- In short, we propose to include the divergence between the observed drawdown distribution of a batch and the synthetic drawdown distribution in the reconstruction loss function.
- The market generator can loosely be interpreted as a moment matching network, focusing on the moments of drawdown rather than returns.

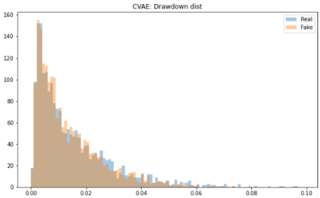
# Market generator: Generated paths



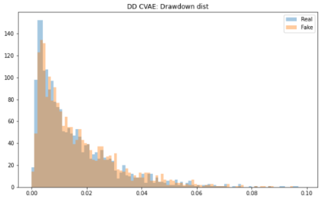

(a) Return dist CVAE




(b) Return dist DD CVAE



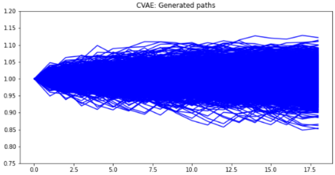

(c) Drawdown dist CVAE



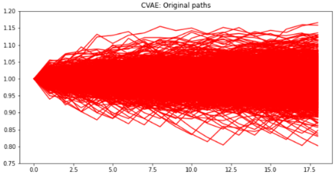

(d) Drawdown dist DD-CVAE

Figure: Generated return and drawdown distributions

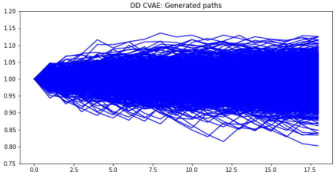
# Market generator: Generated paths



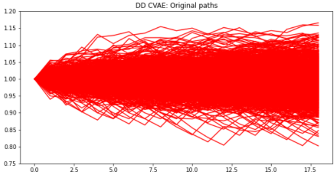

(a) Generated CVAE




(b) Original



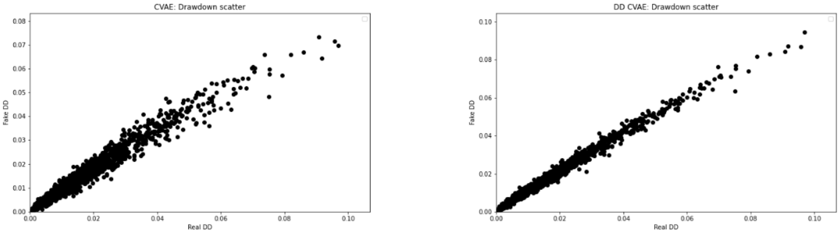

(c) Generated DD-CVAE




(d) Original

Figure: Generated samples

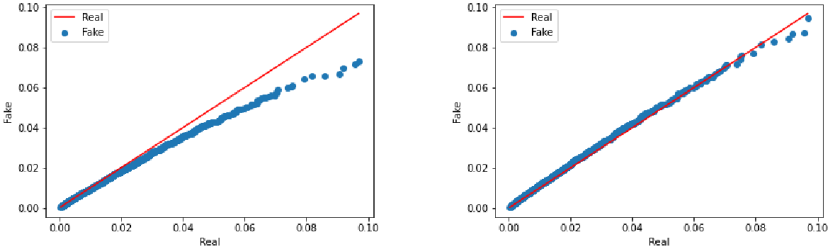
# Market generator: Drawdown scatter




(a) CVAE (b) DD-CVAE

Figure: Drawdown scatter

# Market generator: Drawdown QQ




(a) CVAE (b) DD-CVAE

Figure: Drawdown QQ plot

# Minimum drawdon portfolio (3SA: step 2)

The minimum drawdown portfolio is the solution to the following linear optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbb{E}(\xi(\mathbf{w})) \\ \text{s.t.} \quad & \xi_{\mathbf{t}} = \mathbf{m_t} - \mathbf{wS}_t \\ & \mathbf{m}_t \geq \mathbf{m}_{t-1} \\ & \mathbf{m}_t \geq \mathbf{wS}_t \\ & \mathbf{wI}^N = 1 \\ & \mathbf{w} \geq 0 \end{aligned} \tag{5}$$

where we minimize the expected drawdown $\xi$ as a function of portfolio weights $\mathbf{w}$. The drawdown $\xi$ is a non-linear function of the portfolio path $\mathbf{P_t} = \mathbf{wS}_t$, $\xi_{\mathbf{t}}$ = $\max(\max_{k<t}(\mathbf{P_k}) - \mathbf{P_t}, 0)$, but can hence be written as a linear problem by instrument variable $\mathbf{m_t}$ which denotes the monotonic growth of the portfolio value $\mathbf{m}_t \geq \mathbf{wS}_t$.

# Minimum drawdown portfolio (3SA: step 2)

- It is crucial to understand that we do not rely on a parametric representation of $S$, need no additional regularity like elliptic assumptions, hence make no assumptions about the data generating process (DGP) underlying $S$. Moreover, the model does not require any hyperparameters such as VaR or ES confidence levels.
- The focal element of (5) is the expectation $\mathbb{E}(\xi)$.

# Multiscenario drawdown

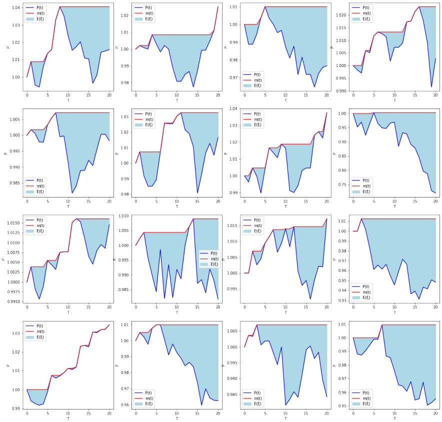


Figure: Example of S for $\tau = 20 < T = 1300$, with $N_S = 1280$ $(= T - \tau)$ overlapping blocks for the US Equity index (S&P500). I.e. multi-scenario drawdown plots for 20 day (+- 1 month) scenarios.

# Model Explainablility (3SA: step 3)

- How does the optimizer respond to shocks?
- We will focus on the US market and build a toy model of the US economy. We collect approximately 100 conditions $C_i$ from the Federal Reserve Economic Data (FRED) database, including credit and monetary data, interest rates, employment, commodity prices, stress indicators, volatility indices, and consumer sentiment.

# Toy macro model of US economy

| Largest positive contributors to $\xi$ | | Largest negative contributors to $\xi$ | |
|---|---|---|---|
| CBOE Volatility Index | 0,815680 | US Gov't Securities at All Com. Banks | -0,142223 |
| Avg Weekly OT Hours: Manufacturing | 0,129751 | Long Term Unemployment: 27 WKS | -0,043401 |
| Exports to Mexico | 0,126585 | JPN/USD Currency Exchange Rate | -0,029723 |
| Univ. of Michigan: Consumer Sentiment | 0,079161 | Avg Hourly Earnings: Manufacturing | -0,001523 |
| St. Louis Financial Stress Index | 0,072814 | | |
| CNY/USD Currency Exchange Rate | 0,068154 | | |
| CAD/USD Currency Exchange Rate | 0,053743 | | |
| Imports from UK | 0,038683 | | |
| 30-yr Conventional Mortgage Rate | 0,037272 | | |
| Effective Federal Funds Rate | 0,029571 | | |

Table: Lasso coefficients of $C_i$ to $\xi$

# Shapley values

Given a set of $n_{cond}$ conditions $C = (C_i)_{i=\{1,\ldots,n_{cond}\}}$, and a set of $N_p$ condition sets $\mathcal{C} = (C^k)_{k=\{1,\ldots,N_p\}}$, each set corresponding to a $C$ that generates $J = n_{cond}$ sequences $S_j$, each C will thus correspond to a unique portfolio that is optimal over this subset of sequences, i.e. for each $k$ one has a different portfolio. The SHAP values to each $w_d^*$ can then formally be defined as:

$$\Phi_i(w_d^*) = \sum_{S \subset [N_s \setminus \{i\}]} \frac{|S|!(N_s - |S| - 1)!}{N_s!} (w_d^*(S \cup \{i\}) - w_d^*(S))) \tag{6}$$

This is the SHAP $\Phi_i$ for condition $C_i$ in terms of the resulting optimal weight $w_d^*$.

# Shapley values (ct'd)

Intuitively, for the $N_p$ optimal portfolios we evaluate all the subsets of $S$ where condition $i$ was not active ($b_j = 0$) and compare with the optimal portfolios where it was $w_d^*(S \cup \{i\})$, or $b_j = 1$. The average contribution of this condition to the optimal weight thus constitutes the SHAP value.

# Shapley values

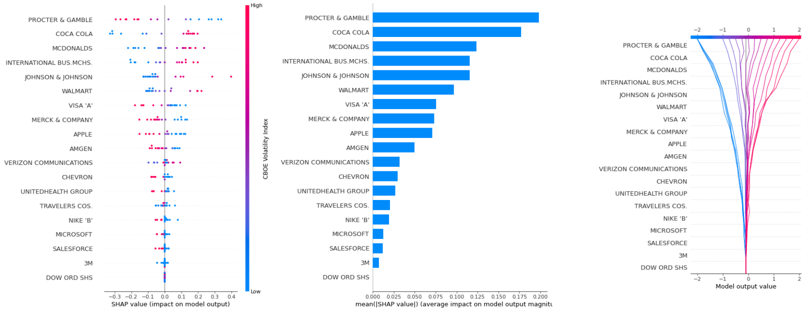



Figure: Shapley values for CBOE Volatility index on the optimal minimum drawdown portfolio. Left: beeswarm plot which indicates the impact on the portfolio weights per instrument and the level of VIX as color. Middle: bar chart which is the average absolute value of the average impact of VIX on the position (SHAP-value). Right: a decision chart or the gradual impact on the portfolio weights by increasing/decreasing VIX.

# Macro dashboard

| | Current Holdings | Univ of Michigan: Consumer Sentiment | US Treasuries Held by the Fed | US Gov't Securities at All Com. Banks | Long Term Unemployment: >27 WKS | JPN/USD Currency Exchange Rate | Exports to Mexico | Civilian Total Unemployment Rate | CNY/USD Currency Exchange Rate | CBOE Volatility Index | Avg Weekly OT Hours: Manufacturing |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Last value | | 59.7 | 5435582 | 4432.808 | 1069 | 129.97 | 25596.236 | 3.5 | 6.782 | 18.51 | 3.6 |
| Previous | | 56.8 | 5436722 | 4439.887 | 1215 | 128.48 | 27915.721 | 3.6 | 6.774 | 18.73 | 3.7 |
| Change | | 5.10% | 0.00% | -0.20% | -12.00% | 1.20% | -8.30% | -2.80% | 0.10% | -1.20% | -2.70% |
| AMGEN | 7.8% | 0.052986258 | 0.052802903 | 0.064200948 | 0.077048502 | 0.056876176 | 0.064109974 | 0.03578008 | 0.030630963 | 0.046127068 | 0.034533547 |
| APPLE | 0.8% | 0.060811318 | 0.048817234 | 0.070668376 | 0.059235012 | 0.075496029 | 0.052866806 | 0.039849038 | 0.034835337 | 0.063317381 | 0.045299441 |
| CHEVRON | 5.3% | 0.05544251 | 0.054247714 | 0.06762688 | 0.039611005 | 0.050165616 | 0.058903097 | 0.053001852 | 0.044836211 | 0.038043243 | 0.047353964 |
| COCA COLA | 5.9% | 0.042207028 | 0.053037406 | 0.059611334 | 0.066422289 | 0.047743565 | 0.079417614 | 0.09266578 | 0.045628382 | 0.180406373 | 0.037008844 |
| JOHNSON & JOHNSON | 10.2% | 0.092732803 | 0.085386992 | 0.082653512 | 0.074167847 | 0.072164284 | 0.122412389 | 0.073350676 | 0.057214875 | 0.121289631 | 0.11907102 |
| MCDONALDS | 9.0% | 0.021489415 | 0.018355678 | 0.042233716 | 0.059096289 | 0.065351926 | 0.041457831 | 0.030147139 | 0.028740571 | 0.119804306 | 0.049674787 |
| MERCK & COMPANY | 6.0% | 0.057017213 | 0.049976877 | 0.057004532 | 0.096488593 | 0.049294659 | 0.08539876 | 0.069782638 | 0.026796147 | 0.076440345 | 0.050035384 |
| MICROSOFT | 3.8% | 0.036461939 | 0.036479167 | 0.047138099 | 0.021903417 | 0.033883353 | 0.026542922 | 0.013347832 | 0.0471462 | 0.01249175 | 0.053637885 |
| NIKE 'B' | 2.3% | 0.031337197 | 0.019478244 | 0.025028338 | 0.024553256 | 0.041202481 | 0.039872278 | 0.031600402 | 0.014046177 | 0.02191613 | 0.028046689 |
| PROCTER & GAMBLE | 13.4% | 0.074973713 | 0.045062375 | 0.050334948 | 0.150795901 | 0.039793467 | 0.081165363 | 0.216342725 | 0.025691296 | 0.202577083 | 0.026208027 |
| TRAVELERS COS. | 4.4% | 0.030693917 | 0.007588248 | 0.012600206 | 0.093026386 | 0.014307226 | 0.012643239 | 0.063997145 | 0.015702092 | 0.01291851 | 0.062671172 |
| UNITEDHEALTH GROUP | 3.5% | 0.036663961 | 0.028029084 | 0.04359054 | 0.027339529 | 0.018009922 | 0.041863718 | 0.03981607 | 0.018623258 | 0.028587745 | 0.035476698 |
| VERIZON COMMUNICATIONS | 8.9% | 0.027278608 | 0.02601565 | 0.02716867 | 0.071636266 | 0.043777217 | 0.032751525 | 0.045683405 | 0.019457459 | 0.032208955 | 0.033845752 |
| VISA 'A' | 0.1% | 0.072777085 | 0.052815853 | 0.060736867 | 0.058056613 | 0.040564638 | 0.073538785 | 0.067579037 | 0.038748602 | 0.074601816 | 0.051777059 |
| WALMART | 8.4% | 0.0246824 | 0.01858237 | 0.053806642 | 0.069252639 | 0.041596232 | 0.054871841 | 0.047000199 | 0.03978451 | 0.098605233 | 0.047263486 |

Figure: Macro dashboard

# Robo trading demo

# Robo trading demo

- At university you don’t often build things.
- Say the analogy is a car, you learn everything about Newtonian mechanics, heat equations, engines, aerodynamics, etc. without ever building even a toy car.

# A *skateboard* robo trading agent

- Quants usually build rockets.
- But to go from A to B one can just start with the minimal functionality such that it still qualifies as a vehicle: a *skateboard*, add a windshield, add a motor, and continuously refine until we have a car.
- This was the initial aim of the demo: an automated trading agent that has at least the modules a real robo trader would have (albeit simplistically).
- Hence disclaimer: *do not share under any pretexts. Use at own responsibility. By no means is this a piece of investment advice.*

# A modular robo trader blueprint

The robo trader blueprint: data integration, signal (*alpha*) extraction, portfolio and risk (*beta*) optimization and order execution (broker integration).

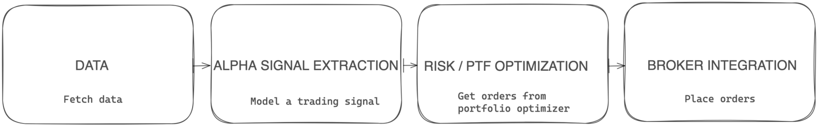



Figure: A modular robo trader blueprint

# IB Trader Workstation (TWS)

Make sure to have account (`https://www.interactivebrokers.ie/en/home.php`) and log in with paper account!

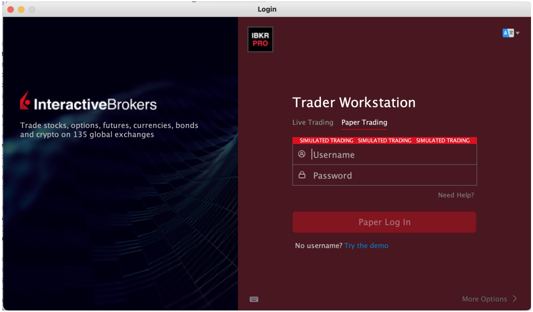
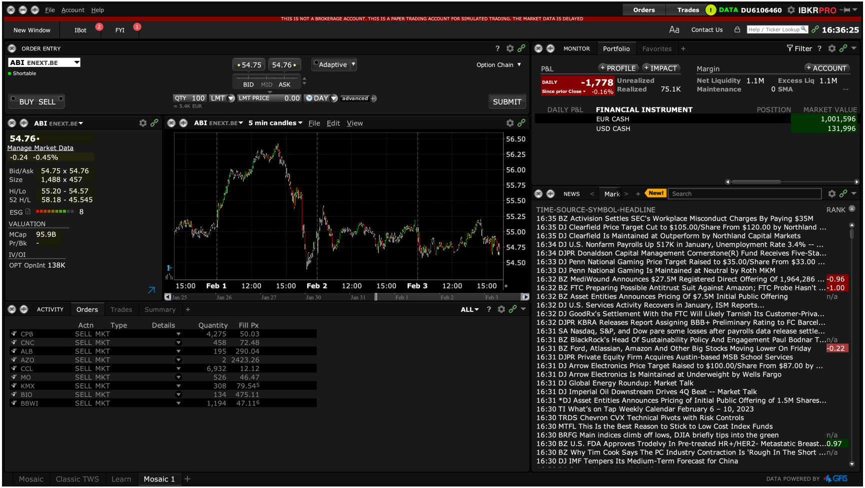



Figure: TWS login with paper account

# IB Trader Workstation (TWS)

## Demo



## 📡💾 Data

[3]:
```
# Ref example
# https://gist.github.com/paduel/32ac6f0a47f3fae67e414a73b9779e89

import yfinance as yf

# Universe size, reduce to speed up optimisations...
n_assets = 100

# Get the current S&P500 components, and get a tickers list
sp_assets = pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')[0]
assets = sp_assets['Symbol'].str.replace('.', '-').tolist()

# Download historical data to a multi-index DataFrame
start = pd.Timestamp("2005")
end = pd.Timestamp.now().date()
data = yf.download(assets[:n_assets], start=start, end=end, as_panel=False)
# Cleaning
# Get close price and filter
ts = data["Adj Close"]
# Filter NaNs and 0 std
nas = ts.isna().sum()
nas = list(nas[nas > 0].index)
std = ts.std()
no_std = list(std[std == 0.0].index)
ts_no_nas = ts.loc[:, [c for c in ts.columns if c not in nas + no_std]]
ts_no_nas.index = pd.to_datetime(ts_no_nas.index)
ts = ts_no_nas

# Define universe
universe = ts.columns
```

```
/var/folders/tj/bzsrq61d07553s4dmx4vvdsm0000gn/T/ipykernel_10019/406957327.py:11: FutureWarning: The default value
*not* be treated as literal strings when regex=True.
  assets = sp_assets['Symbol'].str.replace('.', '-').tolist()
[*********************100%***********************]  100 of 100 completed
```

# Data

```
# Data
ts
```

| Date | A | AAP | AAPL | ABC | ABT | ACGL | ACN | ADBE | ADI | ADM | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2004-12-31 | 14.728202 | 26.124023 | 0.980280 | 11.365149 | 14.580822 | 4.300000 | 19.677843 | 31.363848 | 23.986349 | 14.800267 | ... |
| 2005-01-03 | 14.593752 | 26.046276 | 0.963385 | 11.262497 | 14.587081 | 4.233333 | 19.218699 | 30.838949 | 23.642004 | 14.568077 | ... |
| 2005-01-04 | 14.208743 | 25.872828 | 0.973279 | 11.173402 | 14.440178 | 4.177778 | 18.766836 | 30.024111 | 22.998819 | 14.408871 | ... |
| 2005-01-05 | 14.202630 | 25.920683 | 0.981803 | 11.152101 | 14.274518 | 4.153333 | 18.693954 | 29.859142 | 23.102772 | 14.163408 | ... |
| 2005-01-06 | 13.890956 | 25.896751 | 0.982564 | 11.247001 | 14.596450 | 4.147778 | 18.526327 | 29.364239 | 23.063793 | 14.455298 | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2023-01-27 | 155.690002 | 147.440002 | 145.929993 | 164.160004 | 109.949997 | 64.059998 | 277.269989 | 370.709991 | 170.830002 | 83.629997 | ... |
| 2023-01-30 | 151.740005 | 149.880005 | 143.000000 | 165.309998 | 109.809998 | 63.990002 | 274.320007 | 363.420013 | 168.929993 | 82.250000 | ... |
| 2023-01-31 | 152.080002 | 152.279999 | 144.289993 | 168.960007 | 110.550003 | 64.349998 | 279.049988 | 370.339996 | 171.470001 | 82.849998 | ... |
| 2023-02-01 | 155.449997 | 156.839996 | 145.429993 | 161.050003 | 111.820000 | 64.180000 | 283.600006 | 383.920013 | 176.589996 | 83.370003 | ... |
| 2023-02-02 | 155.500000 | 155.240005 | 150.820007 | 158.470001 | 112.250000 | 61.349998 | 294.100006 | 392.230011 | 179.300003 | 82.309998 | ... |

Figure: Timeseries data

# Optimizer

### 🧠 Optimizer and Backtest

Optimizer class

[8]:
```python
class Optimizer():

    def __init__(self, ts, curr_date):

        # Settings
        self.cov_method = CovarianceMethod.HISTORICAL
        self.window = pd.Timedelta(days = 2 * 365)
        self.vol_w = 1e-1
        self.risk_w = 1
        self.tol = 1e-3
        self.max_w = 1.0
        self.min_w = 0.0
        self.cvar_confidence_level = 0.95

        # Current data at curr_date
        current_ts = ts.loc[curr_date-self.window:curr_date, :]
        # Current returns at curr_date
        self.rets = current_ts.pct_change().dropna()
        # Current volatilities at curr_date
        self.curr_vol = np.sqrt(np.diag(get_covariance_matrix(current_ts, curr_date, self.cov_method)))


    def vol_obj(self, w):
        return self.vol_w * w.dot(self.curr_vol)

    def cvar(self, path):
        return self.risk_w * -np.mean(path.sort_values()[:int(np.floor(len(path)*(1-self.cvar_confidence_level)))])

    def risk_obj(self, w):
        return self.cvar(self.rets.dot(w))

    def lin_obj(self, w):
        return self.risk_obj(w) - self.vol_obj(w)

    def optimize(self):
        # Start weights
        w = w_equal = np.ones(len(universe)) * (1/len(universe))
        # Sum to 1
        linear_constraint = LinearConstraint([np.ones_like(w)], 1, 1)
        # Min and Max weights
        bounds = Bounds(np.ones_like(w) * self.min_w, np.ones_like(w) * self.max_w)
        res = minimize(self.lin_obj,
                       w,
                       method='SLSQP',
                       constraints=[linear_constraint],
                       bounds=bounds
                      )
        return {universe[i]: w for i, w in enumerate(res.x) if w > self.tol}, res.x
```

# Backtester

Backtest

```
%%time

# Optimisation dates
n_years = 5
rebalancing_frequency = "BQ"
optimisation_dates = pd.date_range(ts.index[-1] - pd.Timedelta(days = n_years * 365 + 1), ts.index[-1], freq=rebalancing_frequency)


ptf_pit = {}
equal_pit = {}
obj_values = {}
ref_values = {}

for curr_date in optimisation_dates:

    print(curr_date.date())

    optimizer = Optimizer(ts, curr_date)
    ptf_pit[curr_date], res_temp = optimizer.optimize()
    equal_pit[curr_date] = {i: 1/len(universe) for i in universe}
    obj_values[curr_date] = {"Risk": optimizer.risk_obj(res_temp), "Vol": optimizer.vol_obj(res_temp)}
    ref_values[curr_date] = {"Risk": optimizer.risk_obj(pd.Series(equal_pit[curr_date])), "Vol": optimizer.vol_obj(pd.Series(equal_pit[curr_date]))}
```

```
2018-03-30
2018-06-29
2018-09-28
2018-12-31
2019-03-29
2019-06-28
2019-09-30
2019-12-31
2020-03-31
2020-06-30
2020-09-30
2020-12-31
2021-03-31
2021-06-30
2021-09-30
2021-12-31
2022-03-31
2022-06-30
2022-09-30
2022-12-30
CPU times: user 1min 38s, sys: 16 s, total: 1min 54s
Wall time: 19.3 s
```

Figure: Historical backtest

# Results

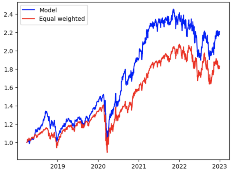



Figure: TimeSeries

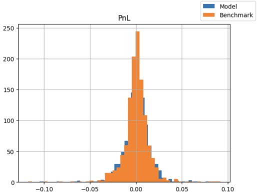



Figure: P&L

# Place orders

```
from ib_insync import *
util.startLoop()

ib = IB()
ib.connect('127.0.0.1', 7497)
```

```
<IB connected to 127.0.0.1:7497 clientId=1>
```

Figure: Connect to TWS

```
# Assuming a certain start portfolio value (IB paper ptf = 1mio EUR)
start_amount = 1e6
# True portfolio value:
field = "AvailableFunds"
summ = ib.accountSummary()
avail_funds = float([f for f in summ if f.tag == field][0].value)
print(f"{field}: {avail_funds}")
```

```
AvailableFunds: 1120947.13
```

Figure: Check available funds

# Place orders

```
# DOUBLE, TRIPLE, QUADRIPLE CHECK whether connected to CORRECT ACCOUNT!
ib.accountSummary()
```

```
[AccountValue(account='DU6106460', tag='AccountType', value='INDIVIDUAL', c
 AccountValue(account='DU6106460', tag='Cushion', value='1', currency='', m
 AccountValue(account='DU6106460', tag='LookAheadNextChange', value='0', cu
 AccountValue(account='DU6106460', tag='AccruedCash', value='-2233.83', cur
```

Figure: Doublecheck whether connected to right (paper) account number!!! When logged in to real-money account we would be placing real trades!

```
orders = []

for position in portfolio:

    # Create a new contract
    contract = Stock(position, "SMART", "USD")

    # TODO : Get live last price, now assumed last know price
    orders += [(contract, MarketOrder('BUY', totalQuantity = int(np.floor(portfolio[position] / last_prices[position]))))]
```

Figure: Organize buy orders

# Place orders

```
# PLACE TRADES: CAREFUL, DOUBLE CHECK ABOVE!!!
import time
trades = {}

for o in orders:

    time.sleep(1)

    trades[o[0].symbol] = ib.placeOrder(o[0], o[1])
```

```
# Check if all orders filled

for t in trades:
    print(trades[t].log[-1])

assert [trades[t].log[-1].message == "Filled" for t in trades]
```

```
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 33, 656826, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 34, 399, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 33, 656826, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 33, 656826, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 33, 656826, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 33, 656826, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 38, 311666, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 35, 33, 656826, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
TradeLogEntry(time=datetime.datetime(2023, 2, 3, 15, 36, 2, 725648, tzinfo=datetime.timezone.utc), status='Filled', message='', errorCode=0)
```

Figure: Place orders and check whether correctly filled

# Place orders

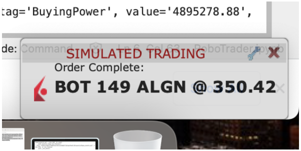
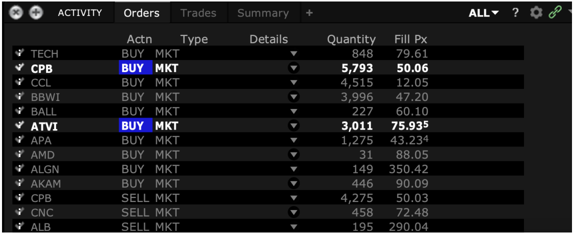



Figure: Order notifications (SIMULATED!)




| | Actn | Type | Details | Quantity | Fill Px |
|---|---|---|---|---|---|
| TECH | BUY | MKT | | 848 | 79.61 |
| CPB | BUY | MKT | | 5,793 | 50.06 |
| CCL | BUY | MKT | | 4,515 | 12.05 |
| BBWI | BUY | MKT | | 3,996 | 47.20 |
| BALL | BUY | MKT | | 227 | 60.10 |
| ATVI | BUY | MKT | | 3,011 | 75.935 |
| APA | BUY | MKT | | 1,275 | 43.234 |
| AMD | BUY | MKT | | 31 | 88.05 |
| ALGN | BUY | MKT | | 149 | 350.42 |
| AKAM | BUY | MKT | | 446 | 90.09 |
| CPB | SELL | MKT | | 4,275 | 50.03 |
| CNC | SELL | MKT | | 458 | 72.48 |
| ALB | SELL | MKT | | 195 | 290.04 |

Figure: Orders will appear on TWS

# Follow up on performance

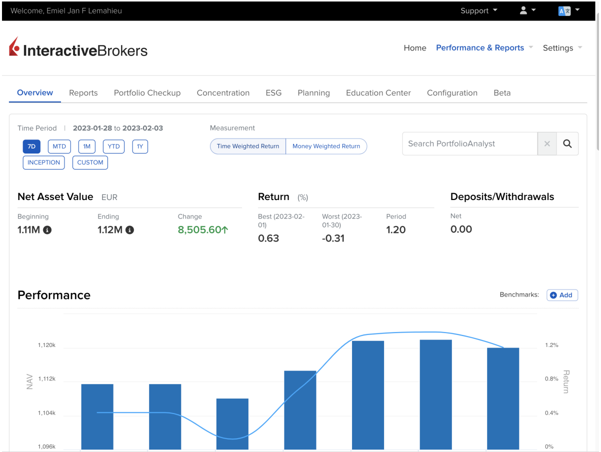



Figure: IB Portfolio Analyst reporting

# Extensions and challenges

- Adding windshields (proper risk management) and engines (alpha)...
- With this skateboard (code) we literally won't get far (return) nor comfortable (risk)...
- Now we have oversimplified modules taking up a couple of cells each, but nevertheless we have all modules to trade.
- In real life, quants build rockets and every module constitutes a package or multiple software packages that make up thousands of lines of code and communicate through APIs.

# Extensions and challenges

In recap, the modules were:

1. Data loading and preprocessing
2. Alpha signal extraction (here assumed vol as opportunity)
3. Portfolio and risk optimization
4. Trading / placing orders at broker

# Extensions and challenges

Each one could be improved:

1. Data loading and preprocessing: looking at multiple sources of market and non-market data.
    - Fundamentals (PE, Net Debt / EBITDA, FCF yield,...),
    - Implied volatility and the vol surface,
    - Analyst recommendations and price target disparity,
    - Liquidity factor,
    - ...

# Extensions and challenges

2. Alpha signal extraction: using the Statistician's complete Toolbox.
   - Linear regression,
   - Trees, random forest, XGBoost,
   - (Recurrent) Neural Networks, LSTM,
   - Signatures
   - ...
   - on numbers and *text* as data!!!

# Extensions and challenges

3. Portfolio and risk optimization: using a more realistic Optimizer.
    - Not using a generic solver, but defining a proper linear(-quadratic) programme.
    - Exploration of volatility, Value-at-Risk (VaR), expected shortfall / Conditional Value-at-Risk (CVaR), or Drawdown as a risk measure
    - Exploration of constrained optimization problem (v.a.v sectors, regions, or a chosen Benchmark (!))
    - Inclusion of regularization (overfitting) or transaction costs (turnover penalty/constraint) objectives.

# Extensions and challenges

4. Trading / placing orders at broker: optimizing order execution.
   - Modeling optimal order execution (order book analysis),
   - Or, more realistically, analyzing impact of Order Types (other than MarketOrder, such as LimitOrder, BracketOrder (LimitOrder with PROFIT_TARGET or STOP_LOSS criteria in OCA (One-Cancels-All) Order Group `https://interactivebrokers.github.io/tws-api/basic_orders.html`)

## Conclusions

1. Roboadvisors automate investment advice.
2. AI automates tasks that require cognition in general.
3. Machine learning learns new rules from data, instead of giving answers on data for known rules.
4. Approximate sparsity (complex world, simple rules) and universal approximation (complex functions, simple approximations) are the core ideas from ML/AI that can guide us towards novel quantitative tools for finance.
5. Portfolio construction and ML/AI has been a difficult marriage, but one has to appreciate where one can use ML/AI to enhance the portfolio construction process rather than replacing it.
6. Our suggestion was the 3-step approach (3SA) of simulation, optimization and attribution.

## Conclusions

7. We first delved into simulation and the elegance of signature approximated drawdown for learning realistic scenarios.
8. Then we optimized on those scenarios and considered the conditions that gave rise to them.
9. Next, we attributed the difference in portfolio to the difference in condition values using Shapley values.
10. Finally, we looked at a skateboard approach to robo trading.
11. We built the simplest possible solution that contained all the modules required to go from data, through optimization and backtesting, to order execution.
12. We concluded with extensions, challenges and next steps to gradually refine the skateboard into a car, potentially into a rocket.

# References I


- Neil E Cotter, *The stone-weierstrass theorem and its application to neural networks*, IEEE transactions on neural networks **1** (1990), no. 4, 290–295.
- Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2, Springer, 2009.
- Diederik P Kingma and Max Welling, *Stochastic gradient vb and the variational auto-encoder*, Second International Conference on Learning Representations, ICLR, vol. 19, 2014, p. 121.

# Thank you!

Questions? Thoughts? Comments? Remarks?