SDSC 4116: Data Science Capstone

Shapley Value Method for Risk and Return Attribution

ZHUMAKHAN Arman

CHAUHAN Kunal Singh

Supervisor: Dr WU Qi

City University of Hong Kong

15 April, 2023

Content

Abstract

Shapley value is the contribution of each player in a cooperative game. By observing the contribution of each player, different inferences can be made. It is widely applicable, such as observing the participation of each stock in a portfolio in their contribution to the profit and loss, or certain risk factors to assess the risk added by including a specific stock.

The application of Shapley Value in portfolio management and assessment in the past has been highly theoretical, most papers have only focused on generated data, rather than testing out their theories on empirical datasets. Therefore, one of our objectives was to test the validity of Portfolio Assessment, and Risk and Return Attribution through Shapley Value on real data and expand the existing validity of the theories. We proved that by making some key changes, existing theoretical research can be applied to real data, with similar, if slightly worse results. This proof opens the possibility of Shapley Value being implemented to solve real-world problems.

Introduction

Shapley Value According to L. S. Shapley et al. (1953), Shapley Value is used to calculate a value for a game with the following assumptions:

- Utility is objective and transferable (Utility being every prospect that can arise from playing)
- The games are a cooperative affair.
- The games granting 1 and 2, are adequately by its characteristic functions.

Please find the formula and explanation of Shapley Value in Theoretical Background.

Example Let us consider a simple voting game with four parties *A, B, C, D* with 45, 25, 15, and 15 representatives, respectively. They are to vote on whether to pass a $100 million spending bill and how much of this amount should be controlled by each of the parties. A majority vote, that is, a minimum of 51 votes, is required to pass any legislation, and if the bill does not pass then every party gets zero to spend. *A, B, C, and D* have 45, 25, 15, and 15 votes. 51 votes are required to pass the $100 million bill. Therefore, A is in all winning coalitions, but doesn't win alone. *B, C, D* are interchangeable as they always provide the same marginal benefit to each coalition. They add $100 million to the coalitions *{B, C}, {C, D}, {B, D}* that do not include them already and to *{A}*. They add $0 to all other coalitions. Grinding through the Shapley value calculation, we get the payoff division (50, 16.66, 16.66, 16.66), which adds up to the entire $100 million.

Applications The original idea of Shapley value is from cooperative game theory (Shapley, 1953). In the original paper, Shapley value is used to allocate the surplus of a coalition to each player. H. Moulin (2004) describes Shapley value as "the single most important contribution of game theory to distributive justice", showing its importance in fair value allocation. Another area, recently gaining more attention, is interpreting the output of machine learning models such as

neural networks or random forests (Strumbelj, 2014). Inputs to models are considered as players in a coalition, and prediction value, y, is a coalition value. This application is widely used and has a lot of open-source projects. For example, SHAP in Python is popular library with about 18,200 stars in GitHub (Lundberg, 2017). Another application of Shapley value is risk decomposition of the investment portfolio of stocks or derivatives (Hagan et al., 2021; Colini-Baldechi et al., 2018). Using different risk metrics such as greeks or variance, we can understand what assets are causing more risk.

Overview In this paper, we explained Shapley Value in detail. We will now discuss all the papers we researched. We will then show the experiments we carried out and our results, how have we improved on the original paper and what more can be done in the future. The report has been formatted as follows, first we will discuss the research papers we have studied, then we will go over all the theoretical background required to understand our experiments. Then we will go over our experiments and results.

Literature Review

Portfolio risk allocation through Shapley value *by P. S. Hagan, A. Lesniewski, G. E. Skoufis, and D. E. Woodward.*

Hagan et al. (2021) argue about the application of Shapley value of cooperative game theory on risk allocation scheme creation. In other words, to naturally interpret the contribution of each risk factor on overall portfolio risk using Shapley value. In addition, they demonstrate a method to interpret enterprise risk metrics such as Value at Risk and Expected Shortfall (VaR and ES).

Attribution of VAR and ES Hagan et al. (2021) used Shapley value to allocate enterprise risk metrics such as VaR and ES to the risk factor of the portfolio. VaR or Value at Risk is a statistic used to quantify the risk of a portfolio. It represents the maximum expected loss with a certain confidence level. ES or Expected Shortfall is a statistic used to quantify the risk of a portfolio. Given a certain confidence level, this measure represents the expected loss when it is greater than the value of the VaR calculated with that confidence level.

Hagan et al. (2021) used Historical Bootstrapping to estimate the values of the Shapley Values of VaR and ES. They also created an estimate for this value and tested how well the estimate performed for a dataset they created.

Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Values Approximation *by M. Ancona, C. Oztireli, M. Gross (2019)*

Objective Deep Neural Networks are a black box model (models which provide useful information without revealing any information about its internal working). Making them more difficult to adopt for tasks where interpretability is a requirement.

Since global interpretability of DNN's makes no sense, in this paper, M. Ancona et al. (2019) decided to focus on local interpretability, which aims to explain a particular decision for a given model and input instance.

To achieve this, they looked at axiomatic approaches (an axiom is a self-evident property of the attribution method that should be satisfied for any explanation generated by the method itself). By leveraging these properties, attribution methods with stronger theoretical guarantees can be designed.

So, they ended up looking into Shapley Value as a unique way of assigning attribution such that certain desired axioms are satisfied.

Computation of the exact Shapley Value is hard and only computationally feasible for 20-25 players. And thus, this paper contributed in the following ways:

- endorsing an axiomatic approach, compare Shapley values to existing state-of-the-art attribution methods, motivating the use of the former to explain non-linear models.
- formulate a novel, polynomial-time approximation of Shapley values specifically designed for DNNs.
- assess empirically the approximation power of our algorithm compared to other attribution methods on three datasets and architectures.

Through this paper, M Ancona et al. (2019) showed that existing attribution models reduce computing Shapley Value when applied to linear models. Meanwhile, when the model is non-linear, Shapley Value is the only method satisfying several desirable theoretical properties.

They recommend some improvements, using DASP for recurrent neural networks, which could lead to new probabilistic frameworks that will enable the derivation of theoretical guarantees and even better approximations.

An Application of Shapley Value Cost Allocation to Liquidity Savings Mechanisms *by Rodney J. Garratt (2019)*

Nowadays, the central bank supervises or provides an infrastructure for large-value payments settlements between banks. Historically, banks used end-of-day netting systems, but with increasing volumes and values, there is an existing risk in deferred payment systems. The most popular now is the Real-Time Gross Settlement (RGTS) system. With RGTS, all payments are settled individually, instantly, and during operational hours. This system ensures no risk of unwinding of payments by increasing requirements for liquidity to be provided by the banks.

In perfect cases, the liquidity provided should be always sufficient to offset the queued payments by the banks whose net obligations are positive. However, in reality, this is not always the case. If there are n payments in the queue, there are 2n-1 possible combinations to be considered. This is computationally too expensive for a real-life scenario. Therefore, in some centralized queuing systems, the minimum number of payments required to settle all the payments is considered. The author explains the way to apply Shapley value for cost allocation problem to settle a set of netted payments.

Let the number of banks be $N = \{1, \ldots, n\}$. Denote P is the 3-dimensional matrix of all the payments in the queue between each bank. Let p_{ijk} of P be a vector of payments between banks i and j, and k be the number of payments with k = 1, 2, … Assume there is a pre-defined cost of providing liquidity by each bank at the current settlement opportunity c_i.

Using this information, banks can assess the instantaneous cost and profit of any net proposal. Using Shapley Value, R. J. Garrat (2019) provided a method to distribute the amount a bank owes among all banks that require side payments based on what each bank is owed, relative to what all the banks that require side payments are owed.

The author made a proposal of a new approach for liquidity saving mechanisms, involving take-it-or-leave-it proposals. The potential of the proposed system's performance relies on the ability to accurately assess the benefits and costs for each bank regarding how to provide liquidity. This is the main downside of the proposed system because in practice, the center does not know instantaneous benefits for any bank as this is classified information. Perhaps, this can be avoided by keeping track of the instantaneous benefits historically gained by each bank from previous payments which can still be up to date.

Portfolio Performance Attribution via Shapley Value *by N. Moehle, S. Boyd, A. Ang. (2021)*

An investment process has some performance measures over a certain period. Some measures are preferred to be large (P&L, return) and others are better to be small such as risk, and turnover. Investment processes employ different features that can be active (turned on), or inactive (turned off). Consequently, we want to know how much of the performance is attributed to each of these features and a baseline. The baseline value is what the performance would have been when all features were off. Positive features mean that the features impacted positively the performance. A feature attribution can be negative, showing that the feature diminished the performance value.

Suppose we have an investment process with some optimization method that use a leverage limit, ESG constraint, and a novel return forecast method. ESG constraint is used to limit trading of certain securities. Suppose our investment process has a return of 8% over one year, with all features active. Attribution may be 1% to the leverage limit, -1% to the ESG constraint, 5% to the return forecast, and 3% to the baseline. We interpret this as each feature contributed to the performance with its corresponding return.

Attribution has different applications ranging from credit allocation for bonus payments to additional cost of features. There are some researches that use Shapley attribution for risk decomposition, but this is the first application of Shapley value to the general portfolio attribution problem. In the paper, attribution is used in explaining performance of portfolio. The paper discusses several attribution methods, namely one-at-a-time, leave-one-out, sequential, permuted sequential, Shapley attributions, and compare their properties.

One of the examples, the paper gives, simulated an investment strategy based on Markowitz portfolio optimization and back tested with real stock data with the benchmark S&P 500 and data

from 2002 to 2019. Authors calculated the attribution of four performance metrics for a tax-aware portfolio management process: realized post-tax return, ex-ante risk, capital gains, and portfolio turnover. Comparing the values of different attribution methods, the paper showed that Shapley value reflected logical attribution and is the superior method compared to one-at-a-time and leave-one-out methods.

The paper concludes that Shapley value is the proposed method for portfolio performance attribution and possesses all necessary properties such as fairness, correct baseline, full attribution and monotonicity. The only disadvantage of Shapley value is expensive computation. It can be solved by using approximation methods. The authors show 3 ways to approximate the Shapley value calculations as well.

Multi-Period Trading via Convex Optimization *by Stephen Boyd, Enzo Busseti, Steven Diamond, Ronald N. Kahn, Kwangmoo Koh, Peter Nystrup and Jan Speth (2016)*

Using an optimization task to choose an investment portfolio started with Markowitz (1952). The initial idea employed risk and return to choose assets without consideration of other costs. When trades are executed over to long periods, other costs add up significantly. Since that, researchers started to include different constraints and many costs and in a single period optimization formula. The portfolio selection task in multi-period portfolio becomes to select trades over several time periods. Though there have been a lot of research on the multi-period portfolio, mostly with dynamic programming, most of them have very few assets to consider or simple constrains. The advent of better algorithms and more powerful computers allows the paper is to conduct multi-period convex optimization with constraints to choose portfolio assets from huge number of assets.

Authors describe single-period optimization (SPO) to choose assets over a single period. The optimization objective includes hyperparameters related to holding cost, transaction cost, and risk, which give more freedom to encourage or discourage more certain constrains.

The paper suggests multi-period optimization (MPO) trading strategy to consider information of the multiple periods. It has several advantages over the SPO. For example, MPO considers transactions costs in the future because current holdings significantly impact the profitability of the strategy. Some return predictions might be positive only in a short period, but not long periods. MPO also helps to avoid risky future positions in advance when the risk is increasing.

Authors simulated strategies with open-source market data of 5 years, from January 2012 to December 2016. Strategies continuous selected trades each month from the components of the

S&P 500 index. The paper conducted multiple backtested with different risk aversion, trading aversion, and holding cost parameters for both SPO and MPO. Each backtest of 5 years of data took only about 5 minutes with daily optimization. It allows portfolio managers to test different strategies within short period of time.

Theoretical Background

Shapley Value Formula

A game is a set of rules with players, and these specified players are in a position to play. A coalition game with transferable utility here means a pair (N, v) where N is the number of players, and for some $S \subset N$. *v(S)* is the real value payoff which the coalition members distribute among themselves, with $v(\Phi) = 0$. Let us look at the formula for Shapley Value to understand how this concept is implemented for a coalition game with (N, v).

$$\phi_i(N, v) = \frac{1}{N!} \sum |S|! \, (|N| - |S| - 1)! \, [v(S \cup \{i\}) - v(S)] \text{ , where } S \subset N \backslash \{i\}$$

This is a calculation of the "average marginal contribution" of agent i. Here, we start by averaging over all the different sequences according to which the grand coalition could be built up from the empty coalition. Here, if agent i is added, its contribution is $[v(S \cup \{i\}) - v(S)]$ we will multiply this quantity by $|S|!$ the different ways the set S could have been formed prior to agent i's addition and by the $(|N| - |S| - 1)!$ different ways the remaining agents could be added afterward Finally, sum over all possible sets S and obtain an average by dividing by $|N|!$, the number of possible orderings of all the agents.

Experiment 1. Performance attribution

We suppose that an investment process produces a dynamic portfolio allocation over certain period. It has *n* features that can be included or excluded (active or inactive). Features is expressed with $x_i \in \{0, 1\}$, with 0 meaning inactive, 1 active. The collection of feature values is a configuration of the investment process. It is denoted by the Boolean vector $x = (x_1, \dots, x_n)$. A number of possible configurations increases exponentially with 2^n rate. In the investment process with n=10, there are 1000 possible configurations; with n=30, there are 10^9 configurations.

The configurations with all features active, $x=(1,1,\dots,1) = 1$ is referred as the *full configuration*. The configuration with all features inactive, $x=(0,\dots,0) = 0$ is called the *zero configuration* or the *baseline configuration*. The investment process is evaluated using a performance metric $y \in \mathbb{R}$.

The paper used simulation to evaluate performance with different configurations. This has the form of a backtesting engine. The process is represented by a function $f: \{0,1\}^n \rightarrow \mathbb{R}$, with

$y = f(x) = f(x_1,\dots, x_n)$.

Suppose there is the configuration x with $x_i = 0$. We refer e_i as the i-th unit vector. Then $x' = x + e_i$ is the configuration obtained by making feature i active. The lift or marginal contribution by adding feature i is $f(x + e_i) - f(x)$. This marginal contribution depends on the given configuration x or which features are already active or not.

Attribution methods

1. **One-at-a-time attribution** is calculated by

$a_i = f(e_i) - f(0), i = 1,\dots, n.$

The attribution of each feature is change in performance when this feature is added to the baseline configuration. This method needs to do only n (n+1 if we include the baseline) simulations.

2. **Leave-one-out attribution** is set by $a_i = f(1) - f(1-e_i)$, $i=1,\ldots, n$. The attribution is the marginal performance change of adding feature *i* when all other features are present. Leave-one-out attribution requires n simulations like one-at-a-time attribution.

3. **Sequential attribution.** We start by calculating the baseline configuration performance b = f(0). Then, we keep adding 1 feature and simulate the configuration until we have all features. The sequential attribution takes a form

$a_i = f(e_1 + \cdots + e_i) - f(e_1 + \cdots + e_{i-1})$, $i=1,\ldots, n$.

The formula is the marginal performance of feature i with the feature 1, ..., i-1 are active. This method requires n simulations like previous two methods.

4. **Permuted sequential attribution.** This is an extension of sequential attribution. We use $\pi = (k_1,\ldots,k_n)$, which is a permutation of (1,...,n). Then, we take the configuration vector when we permute the features using π and apply sequential attribution on the new order of features. In the sequential attribution, we add features one by one in order, whereas in the permuted sequential attribution, we add features in the order of $(k_1,\ldots,k_n)$.

5. **Shapley attribution**

Shapley attribution can be regarded as the average of the permuted sequential attributions over all n! permutations. Let's denote a_π the attribution for permuted sequential attribution with permutation π. Shapley value becomes

$a = \frac{1}{n!} \sum_\pi a_\pi$, where the sum is over all n! permutations.

Experiment 2. Portfolio Management based on Convex Optimization

Model

Portfolio. Portfolio consists of holdings in *n* assets and a cash account. Portfolio is updated every discrete time periods labelled t = 1,…,T. These time periods are arbitrary, could be daily, weekly, or hourly, for example. $h_t \in \mathbb{R}^n$ is used to denote the portfolio the portfolio holdings in the dollar value at the start of the period t. The value of $(h_t)_{n+1}$ is the cash balance. If $(h_t)_{n+1} = 0$, the portfolio is considered as fully invested showing that we hold no cash.

The total value of the portfolio (in dollars) is $v_t = \mathbf{1}^T h_t$. The gross exposure of the portfolio can be calculated by $\|(h_t)_{1:n}\|_1 = |(h_t)_1| + \cdots + |(h_t)_n|$, which is the sum of the absolute values of the positions. The leverage of the portfolio is equal to the gross exposure divided by the total value, $\frac{\|(h_t)_{1:n}\|_1}{v_t}$.

In some cases, it's easier to use weights of assets in terms of the total value. Weights $w_t \in \mathbb{R}^n$ are calculated by $w_t = \frac{h_t}{v_t}$. The weights sum to one. The leverage of the portfolio is equal to the L1 norm of the asset weights, $\|w_{1:n}\|_1$.

Trades. We used $u_t \in \mathbb{R}^n$ to denote the trades in dollars. Positive values $(u_t)_i > 0$ means that we buy asset *i* and negative values $(u_t)_i < 0$ means we sell asset *i* at the period of t, for *i* = 1,…,n. The vector $z_t = u_t/v_t$ is the normalized trades by the total value and unitless. Half of its L1 norm $\|(u_t)_{1:n}\|_1/2$ is the turnover in dollars at the period t. Turnover can be expressed in a percentage as $\|(u_t)_{1:n}\|_1/(2v_t) = \|z_{1:n}\|_1/2$.

The post-trade portfolio is given as

$$h_t^+ = h_t + u_t$$

Its means the portfolio holding immediately after trading at the period t.

Transaction cost. Every time, asset is sold or bought, a trading or transaction cost occurs, which is denoted as $\varphi_t^{trade}(u_t)$. Transaction cost function is φ_t^{trade}: $\mathbb{R}^{n+1} \to \mathbb{R}$. It's assumed that there

is no transaction cost with the cash account. Another assumption is that when there is no trade, transaction cost becomes zero $\varphi_t^{trade}(0) = 0$.

A model for the transaction cost function is

$$x \to a|x| + b\sigma \frac{|x|^{3/2}}{V^{1/2}} + cx$$

where *x* is a dollar trade amount, *a* is one half the bid-ask spread for the asset (in fraction of asset price), *b* is a positive constant, *V* is the total market volume for the asset in the given period (in dollars), σ is the price volatility, *c* is a number to create asymmetry in the cost. If $c = 0$, buying and selling the asset have the same transaction cost.

Holding cost. Holding the post-trade portfolio h_t^+ at the period *t* bears a holding cost (in dollars) $\varphi_t^{hold}(h_t^+)$, where $\varphi_t^{hold}: \mathbb{R}^{n+1} \to \mathbb{R}$ is the holding cost function.

A popular holding cost model is charging for borrowing assets when shorting them in the form

$$\varphi_t^{hold}(h_t^+) = s_t^T (h_t^+)_-$$

Where $(s_t)_i \geq 0$ is the borrowing fee for shorting asset *i*, and $(z)_- = \max\{-z, 0\}$ means the negative part of a number *z*.

Self-financing condition. It's assumed that no cash from outside can be put or taken out of the portfolio, and trading and holding costs are settled from the cash account at the start of each period. The self-financing condition can be written as $\mathbf{1}^T u_t + \varphi_t^{trade}(u_t) + \varphi_t^{hold}(h_t^+) = 0$.

Investment. The post-trade portfolio and cash are invested for one period, until the beginning of the next period. After one period, the portfolio becomes

$h_{t+1} = h_t^+ + r_t \circ h_t^+ = (1 + r_t) \circ h_t^+, \quad t = 1,\ldots,\text{T-1},$

where $r_t \in \mathbb{R}^n$ is the vector of asset and cash returns at period t to period t-1, and $\circ$ means elementwise multiplication. The number $(r_t)_{n+1}$ can be the risk-free interest rate.

Metrics

Measuring the portfolio performance against a benchmark is commonly used. The benchmark's weights $w_t^b \in \mathbb{R}^{n+1}$ are share of the assets. Its return is $R_t^b = r_t^T w_t^b$. The active return of the portfolio can be calculated as $R_t^a = R_t^p - R_t^b$. Annualized return of the portfolio is $\sqrt[n]{R_{period}^p}$, where R_{period}^p is the return over a certain period and *n* is period time in years.

The information ratio (IR) of the portfolio is the average of the active returns divided by the standard deviation of the active returns $IR = \overline{R^a}/\sigma^a$

Single-Period Optimization (SPO)

We formulate the problem as convex optimization problem that considers the portfolio performance over one period, and constraints of the portfolio. In the development of a trading strategy, most attention is paid to the formation of estimates or forecasts of the return r_t. The paper considers that estimates are given.

We can express the estimated portfolio return as

$$\hat{R}_t^p = \hat{r}_t^T w_t + \hat{r}_t^T z_t - \varphi_t^{trade}(z_t) - \varphi_t^{hold}(w_t + z_t)$$

Risk-return optimization

To determine the normalized asset trades z_t , we need to solve this optimization problem

$$maximize \quad \hat{R}_t^p - \gamma_t \psi_t(w_t + z_t)$$

$$subject\ to \quad z_t \in Z_t, \qquad w_t + z_t \in W_t$$

$$\mathbf{1}^T z_t + \varphi_t^{trade}(z_t) + \varphi_t^{hold}(w_t + z_t) = 0$$

where $\psi_t: \mathbb{R}^{n+1} \to \mathbb{R}$ is a risk function and γ_t is the risk aversion parameter.

We can replace the self-financing constraint with the approximate constraint $\mathbf{1}^T z_t = 0$. Also, by replacing $\hat{R}_t^p$ with its definition and removing the constant part, we get

$$maximize \quad \hat{r}_t^T z_t - \varphi_t^{trade}(z_t) - \varphi_t^{hold}(w_t + z_t) - \gamma_t \psi_t(w_t + z_t)$$

$$subject\ to \quad \mathbf{1}^T z_t = 0, \qquad z_t \in Z_t, \qquad w_t + z_t \in W_t$$

Risk measures

The risk function ψ_t can be any risk measure of a portfolio. Variance of returns was commonly used traditionally.

Absolute risk. With the assumption that returns are stochastic and Σ_t is the covariance matrix, the variance of R_t^p is

$$var(R_t^p) = (w_t + z_t)^T \, \Sigma_t \, (w_t + z_t)$$

The quadratic risk measure is

$$\psi_t(x) = x^T \, \Sigma_t \, x$$

Active risk. With the assumption that returns are stochastic, the variance of R_t^p is

$$var(R_t^p) = \left(w_t + z_t - w_t^b\right)^T \Sigma_t \, (w_t + z_t - w_t^b)$$

The quadratic risk measure is

$$\psi_t(x) = (x^T - w_t^b) \, \Sigma_t \, (x - w_t^b)$$

Using SPO

We can scale the transaction cost and holding cost rates by aversion parameters $\gamma^{trade}, \gamma^{hold}$ respectively. Changing the trading aversion parameters will help to increase or reduce turnover.

$$maximize \quad \hat{r}_t^T z_t - \gamma^{trade} \varphi_t^{trade}(z_t) - \gamma^{hold} \varphi_t^{hold}(w_t + z_t) - \gamma^{risk} \psi_t(w_t + z_t)$$

$$subject\ to \quad \mathbf{1}^T z_t = 0, \qquad z_t \in Z_t, \qquad w_t + z_t \in W_t$$

Values of hyperparameters $\gamma^{trade}, \gamma^{hold}, \gamma^{risk}$ make a significant change on performance of the SPO. They are usually chosen by back-testing.

Experiment 3. Shapley Value Attribution for VaR and ES

Shapley Value Estimation Proof

For Shapley value attribution of VaR and ES, we will follow the calculations done by Hagan et al. (2021). Let *X* represent the portfolio's daily *P&L*. Under an elliptical distribution, VaR_q and ES_q are linear in the standard deviation $\sigma(X) = \sqrt{\mathrm{Var}(X)}$

$$\mathrm{VaR}_q(X) = \mu(X) - A_q\sigma(X)$$

$$\mathrm{ES}_q(X) = \mu(X) - B_q\sigma(X)$$

Here, A_q and B_q are constants depending only on the distribution and confidence level, and $\mu(X)$ is linear in X. The authors then assume a Multinomial Gaussian Distribution and according to some of their older calculation.

$$VaR_q\,(X) \;=\; \mu^T X \;-\; \Phi^{-1}(q)\sigma(X)$$

$$ES_q\,(X) \;=\; \mu^T X \;-\; \frac{\varphi(\Phi^{-1}(q))}{1-q}\,\sigma(X)$$

Here, $\Phi(z)$ is the cumulative standard normal distribution and $\varphi(z)$ is the density of $\Phi(z)$. Using Shapley value of linear cooperative game, they estimate that $Sh_i(X) = \mu(X_i)$, through approximation, we have.

$$Sh_i(VaR_q) \;\approx\; \mu(X_i\,) \;-\; A_q\rho(X_i\,,X)\sigma(X_i)$$

$$Sh_i(ES_q) \;\approx\; \mu(X_i\,) \;-\; B_q\rho(X_i\,,X)\sigma(X_i)$$

Explanation of Estimator

The focus here is on $A_q\rho(X_i\,,X)\sigma(X_i)$, and $B_q\rho(X_i\,,X)\sigma(X_i)$, where $\rho(X_i\,,X)$ is the covariance of X_i and X, $\sigma(X_i)$ is the standard deviation of X_i, A_q is $\Phi^{-1}(q)$, the z-score for the given distribution and confidence interval q, and B_q is $\frac{\varphi(\Phi^{-1}(q))}{1-q}$, where $\varphi(z) = \;(2\pi)^{-\frac{1}{2}}\; exp(\frac{-z^2}{2})$

Experimental Study

Table 1. Data for the return attribution example in experiment 1

	Benchmark (0, 0)	Country allocation (1, 0)	Stock selection (0, 1)	Full portfolio (1, 1)
f (x)	6.8	4.8	9.4	8.3

Experiment 1. Simple Return Attribution

We will experiment using a simple return attribution example. Metric f(x) is the portfolio return in percentage, using the strategy with selected features. Feature 1 is the country allocation strategy, feature 2 is the stock selection strategy. Table 5 shows example data for the experiment. We applied 3 different attribution methods, namely one-at-a-time, sequential, and Shapley, discussed in Moehle's research (2021) in the literature review part.

The result of the attribution methods is shown in Table 6. Code is in the appendix. One-at-a-time method has a non-zero unattributed return because it doesn't have full attribution property. Sequential method has all unattributed return in the stock selection because it doesn't treat features equally by prioritizing the country allocation over the stock selection strategy. The Shapley method does not have attributed return and treats all features equally. It can be seen that unattributed return of the one-at-a-time is split into 2 features equally in the Shapley method. We can conclude that Shapley method is fair and has full attribution.

Table 2. Attribution results for experiment 1

	Benchmark b	Country allocation a_1	Stock selection a_2	Unattributed $y - a_1 - a_2 - b$
One at a time	6.8	-2.0	2.6	0.9
Sequential	6.8	-2.0	3.5	0
Shapley	6.8	-1.55	3.05	0

Experiment 2. Portfolio management via Convex Optimization

We carried out the single portfolio optimization problem on the market data using CVXPortfolio open-source code in Python (Boyd, 2016). We used annualized returns of the backtest performance as a metric to compare attribution methods.

Trading strategy. We simulated an investment strategy similar to Boyd (2016), formulating the single portfolio optimization. We changed the meaning of hyperparameters $\gamma^{trade}, \gamma^{hold}, \gamma^{risk}$ to be binary variables (1 or 0), so that we can turn on or off features. So, we have 3 features in this configuration. It allows to apply attribution methods on the metrics. The only difference of the current optimization problem to the one we discussed in the theoretical part is we multiplied constant 5 to the risk measure to focus more on the risk management.

$$maximize \quad \hat{r}_t^T z_t - \gamma^{trade}\varphi_t^{trade}(z_t) - \gamma^{hold}\varphi_t^{hold}(w_t + z_t) - \gamma^{risk}\psi_t(w_t + z_t)$$

$$subject\ to \quad \mathbf{1}^T z_t = 0, \qquad z_t \in Z_t, \qquad w_t + z_t \in W_t$$

When all 3 features are 0, the portfolio makes a decision based on the estimated returns.

Backtests. All our simulations use data over a period of 5 years, from January 2017 to December 2022, on the components of the S&P 500 index as of January 2023. We collected the open-source market data from Yahoo Finance API. The data consists of the realized daily price data and volumes. The federal reserve overnight rate was used as the cash return. In holding costs of all assets, we used $s_t = 0.01\%$ (1 basis point). The transaction cost function used $a_t = 0.05\%$ (5 basis points). Values of other parameters in cost models $b_t = 1, c_t = 0, d_t = 0$.

The starting allocation of the portfolio for all simulations was uniform distribution $w_1 = \left(\frac{\mathbf{1}}{n}, 0\right)$.

We set the leverage constraint to 1. Return forecast for simple simulation was generated by means of returns with rolling window of 250 days. The return estimate formula $(\widehat{r_t})_i = \frac{1}{250}\sum_{\tau=1}^{250}(r_{t-\tau})_i$

Though it's not a real return prediction, it's enough to illustrate the idea. We similarly estimated the volatility with variance with window of length 250 days.

Results

We carried out multiple backtests with different hyperparameter combinations ($\gamma^{trade}, \gamma^{hold}, \gamma^{risk}$). We show the some of the backtest performances to look at the effects of features. When all features are turned on (or all hyperparameters are 1), the annualized return of the portfolio was 17.82%. S&P 500 index showed only 7.20% annualized return over the same period. Its performance compared to the benchmark S&P 500 index is given in the Figure 1.

Figure 1. Performance of portfolio using SPO (1,1,1) and S&P 500 index

Weights of the portfolio SPO (1,1,1) is shown in Figure 2. The red line is Apple's stock (AAPL). Legend of the plot colors isn't displayed because a large number of stocks (about 500) didn't fit the graph.

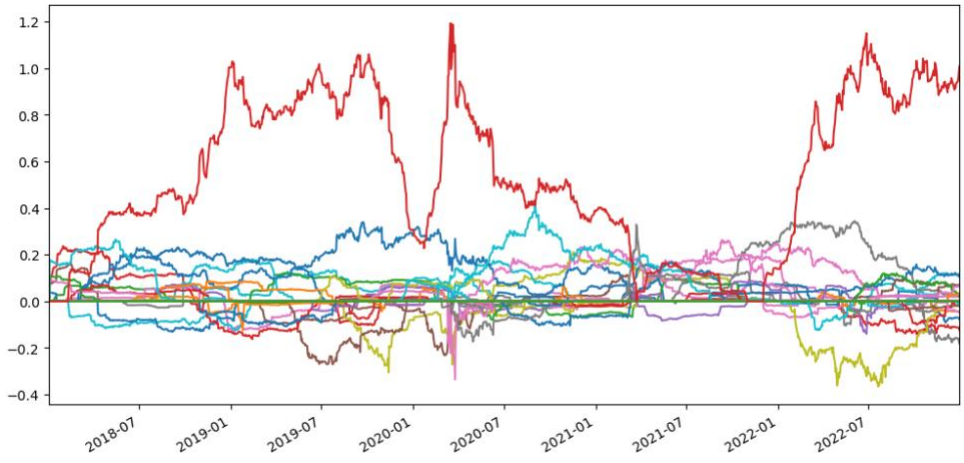


Figure 2. Weights of assets in the portfolio using SPO (1,1,1)

The risk metric in the optimization plays a key role to avert the high volatility, which is our risk metric. When the risk feature is turned off or $\gamma^{risk} = 0$, the portfolio gets much higher returns. It reached the annualized return of 66.27%. Its volatility (risk) increased with it. This is the expected outcome of the trading. The backtest performance is given in the Figure 3.

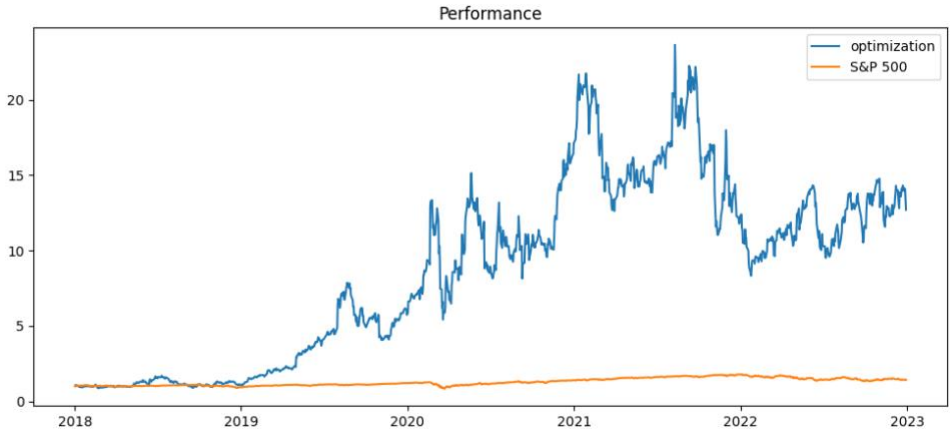


Figure 3. The performance of the portfolio with SPO (1,1,0)

When the feature of trade cost is turned off from the optimization problem or $\gamma^{trade} = 0$, we expect the portfolio to make more trades and lose more money on transactions be default. As anticipated, the portfolio with SPO (0,1,1), or $\gamma^{trade} = 0$, had the annualized return of 11.67%,

less than the portfolio with all features (17.82%). Also, it made much more trades than previous ones. Weights of that portfolio in Figure 4 shows that it made a lot of changes to the portfolio.

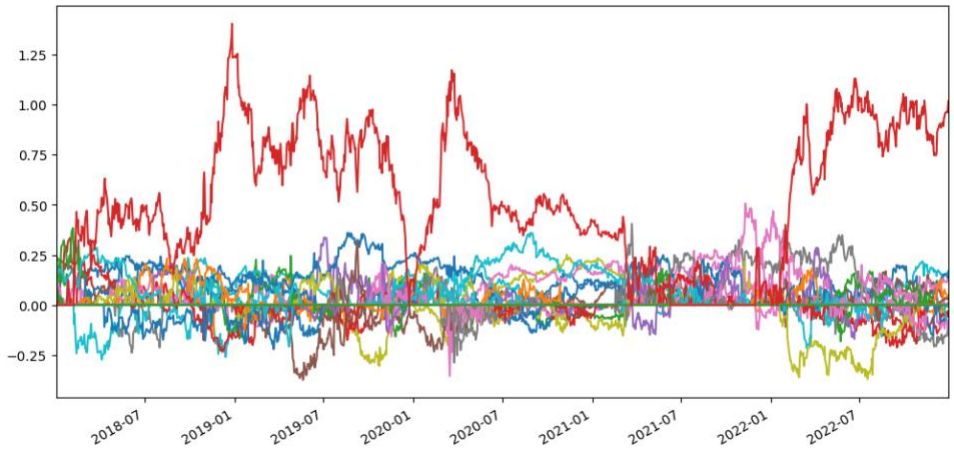


Figure 4. Weights of assets in the portfolio using SPO (0,1,1)

As we have 3 binary hyperparameters, we get 8 different combinations to carry out backtests. We collected all the backtest results in Table 3.

Table 3. Backtest results with different hyperparameters

Number	γ^{trade}	γ^{hold}	γ^{risk}	*f(x)*, return, %
1	0	0	0	56.36
2	1	1	1	17.84
3	1	1	0	66.27
4	1	0	1	17.52
5	0	1	1	11.67
6	1	0	0	66.27
7	0	1	0	75.49
8	0	0	1	11.69

Using the simulation results, we calculated performance attributions of features (b = 56.36%)

Table 4. Attribution values of features

	Trading cost a_1	Holding cost a_2	Risk measure a_3	Unattributed $y - a_1 - a_2 - a_3 - b$
One at a time	9.91	19.13	-44.67	-22.89
Sequential	9.91	0	-48.43	0
Leave one out	6.17	0.32	-48.43	3.42
Shapley	6.325	6.48	-50.63	0

We see that the one-at-a-time attribution overestimates the contribution of each feature because when only one feature is included, it leads to bigger performance. One the other side, the leave-one-out attribution underestimates contributions of features because one new feature makes a small change when all others are added. These problems seem to be resolved by Shapley attribution. For example, the attribution value of holding cost feature is 19.13 in one-at-a-time, 0 in sequential, 0.32 in leave-one-out methods, 6.48 in Shapley. Values of Shapley attribution seem more reasonable because it takes the average change over different combinations.

We suggest using the Shapley value for performance attribution. Shapley values have all necessary properties: fairness, full attribution, monotonicity. When using Shapley value in the setting of a few parameters, the computational cost of it is not high.

Experiment 3. Shapley Value Attribution for VaR and ES

Introduction

Based on the experiments by Hagan et al. (2021), we decided to first recreate their results for Shapley Value estimation for VaR and ES. We then applied the same to empirical data based on the S&P 500 index, to observe how well the estimation works on real data. We then applied it to multiple periods to see the difference between them and try to explain the changes. Finally, we also did similar experiments on the Hang Seng Index to see what inferences we could make.

Accuracy of Estimation Method

We start off by creating a dataset with the same properties as Hagan et al. (2021). We first generated a 25x25 Positive Definitive Matrix and then generated 500 independent samples from the multivariate gaussian distribution. This results in a dataset with a shape of 25x500, representing in our experiment, a set of 25 companies and the stock price data of their last 500 days.

The next step was calculating the estimator for the VaR and ES as provided by Hagan et al. (2021). We first calculated the value of the estimator. Then we had to calculate the Shapley (VaR) and the Shapley (ES), to do this we used historical bootstrapping and Monte Carlo Simulations. The Python3 code for Shapley Value calculation using Monte Carlo Simulation can be found in the appendix (A1). The Python3 code snippet explaining the calculation of VaR and ES as char_fct or character function for the calculation of Shapley Value can also be found in the appendix (A2).

We will explain the concept of the code briefly. We first take the total set of N players (here 25), then, for each player P_i we calculate the Shapley value. For M simulations (we used 500), in each simulation a random number of players are chosen, then for each simulation we see the results with or without P_i and Shapley value is the difference between the two values. To calculate the VaR and ES, for every day, we looked at all the players included and calculated

their daily loss or profit. Then, for every day, we took the biggest loss on that day and stored it. After the code is done, we have 500 days maximum losses, we then sort these losses and using the confidence interval q, for VaR we take the $(1-q)^{th}$ value, for confidence interval $q = 95\%$, and 500 days, we will take the $(0.05)x(500) = 25^{th}$ value. As for ES, we take the average of the first $(1-q)^{th}$ values, here we take the average of 25 worst losses.

Table 3. Results for Shapley Attribution on theoretical dataset

Player	Shapley_VaR	Estimated_VaR	Shapley_ES	Estimated_ES
1	-4.1644	-4.7397	-4.9068	-5.8363
2	-1.9451	-2.2202	-2.5152	-2.7146
3	-1.9793	-2.3635	-2.2061	-2.2790
4	-9.3318	-11.9719	-11.3646	-11.7400
5	-3.4561	-3.6948	-4.4542	-4.9643
6	-4.1147	-2.9371	-6.5737	-7.0805
7	-2.8867	-3.3045	-4.0706	-4.1244
8	-4.1089	-4.1630	-5.1515	-4.2097
9	-4.0773	-3.3932	-5.5215	-6.1053
10	-7.6393	-6.5335	-8.3879	-9.4410
11	-3.3836	-3.6841	-4.1107	-4.9437
12	-19.6535	-21.5931	-27.1609	-31.8871
13	-10.7713	-9.9563	-10.4680	-10.7909
14	-3.2609	-3.4648	-5.6493	-6.4333
15	-2.5075	-2.6807	-4.3648	-5.2781
16	-3.7637	-3.4170	-4.7309	-4.5433
17	-6.4030	-5.2655	-8.7133	-9.1931
18	-71.0939	-85.1265	-84.2088	-90.8861
19	-18.3007	-13.1232	-20.2986	-23.7860
20	-24.8744	-21.3557	-34.3480	-30.2626
21	-7.5904	-5.7176	-8.1424	-9.1467

22	-23.3313	-26.7084	-31.3689	-24.8740
23	-25.5435	-20.2496	-30.7224	-35.0533
24	-29.1230	-28.5482	-36.6052	-36.3633
25	-36.9139	-32.5422	-45.5160	-35.5907

As can be observed, all results are in the same order of magnitude, there are some exceptions, but the overall estimation is quite accurate. The correlation calculates using Spearman Correlation from the scipy library for VaR and ES respectively is.

SpearmanrResult(correlation=0.9454, p-value=1.099e-12)

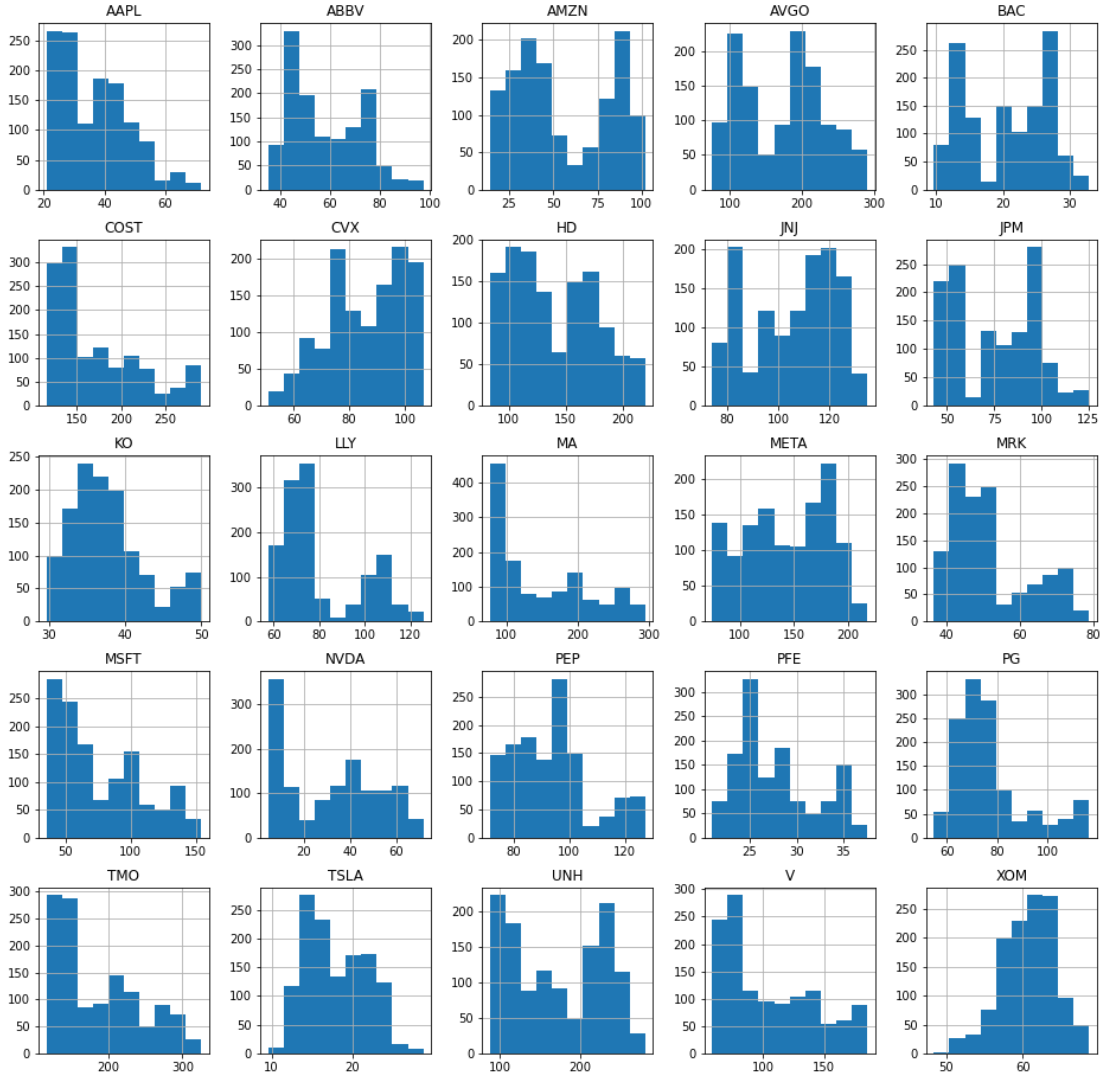
SpearmanrResult(correlation=0.9785, p-value=2.916e-17)

As can be observed, the results are highly correlated, thus we can conclude that the estimator is a good calculator for the Shapley Value.

Empirical Testing

To expand the application of the experiment, we decided to apply the calculations to real data. We got their daily historical data from 2000-2020 using Yahoo Finance. We decided to use the top 25 large cap S&P 500 stocks, the reason we decided to do this was because this would provide us with a wide market breadth of large cap companies. It would also provide us with companies from different industries, reducing market or industry specific phenomena. We ended up with 7 companies related to Technology, 7 companies related to healthcare and pharmaceuticals, 5 companies related to goods and services, 4 related to banking and payment services, and 2 oil companies. We have added below a histogram of the 25 companies with their tickers, to find the associated names with the tickers, please look in the appendix (A3).

Figure 5. Histogram to show the distribution of the stocks



The summary statistics of the data can be found in the appendix (A4).

One important thing we added at this point was that, for the recreation of the experiment we made the data, therefore we know the distribution of the data, but since we are now using empirical data, we need to figure out the closest elliptical distribution for each stock. The reason we figure out elliptical distribution is because the calculations done by Hagan et al. (2021) were only applicable to elliptical distributions. So, we used the Fitter function from the fitter library in Python3 to figure out which distribution best fit each stock and altered A_q and B_q accordingly.

We ran our code to calculate the Shapley (VaR) and Shapley (ES) for four five-year time periods. Find below the results of the calculations from 2015-2020, Table 4-6 containing the results from 2000-2015 can be found in the appendix (A5).

Table 7. Empirical results for Shapley Attribution from 2015-2020

Ticker	Shapley_VaR	Estimated_VaR	Shapley_ES	Estimated_ES
AAPL	-0.0431	-0.0620	-0.0601	-0.0780
ABBV	-0.1551	-0.1232	-0.2739	-0.2543
AMZN	-0.1311	-0.1171	-0.2112	-0.1716
AVGO	-2.0016	-2.6107	-2.7425	-2.6519
BAC	-0.0364	-0.0229	-0.0264	-0.0291
COST	-0.6615	-0.3786	-1.0798	-1.3058
CVX	-0.2229	-0.3348	-0.2561	-0.3449
HD	-0.4496	-0.5828	-0.6444	-0.7189
JNJ	-0.2153	-0.1728	-0.3074	-0.2583
JPM	-0.1835	-0.3133	-0.1800	-0.0958
KO	-0.0486	-0.0374	-0.0496	-0.0264
LLY	-0.3080	-0.0952	-0.3906	-0.4681
MA	-1.0225	-1.4850	-1.7792	-1.9066
META	-1.2311	-1.3222	-2.4182	-1.3529
MRK	-0.1111	-0.0526	-0.1192	-0.0910
MSFT	-0.2764	-0.4945	-0.4478	-0.3073
NVDA	-0.3055	-0.3231	-0.5396	-0.7476
PEP	-0.2123	-0.2630	-0.3240	-0.4935
PFE	-0.0663	-0.0260	-0.1221	-0.1322
PG	-0.2832	-0.1248	-0.4645	-0.4594
TMO	-2.3663	-2.3663	-3.2829	-2.7225
TSLA	-0.2251	-0.1603	-0.3582	-0.3247
UNH	-2.4846	-2.7707	-3.9197	-4.9341

V	-1.6312	-1.4836	-2.3649	-2.0784
XOM	-1.1871	-0.7868	-1.6654	-2.4816

In the above data, if the values are 0, that indicates that the company was not listed on the stock exchange during that period.

The way to interpret these values is that the more negative the values, the more that stock contributes to the VaR or the ES. Therefore, companies with less negative VaR and ES would perform better and lead to less risky portfolios.

Below are some graphs depicting the results and observable changes for the companies.

Figure 6. Change in Shapley (VaR) for the 25 tickers

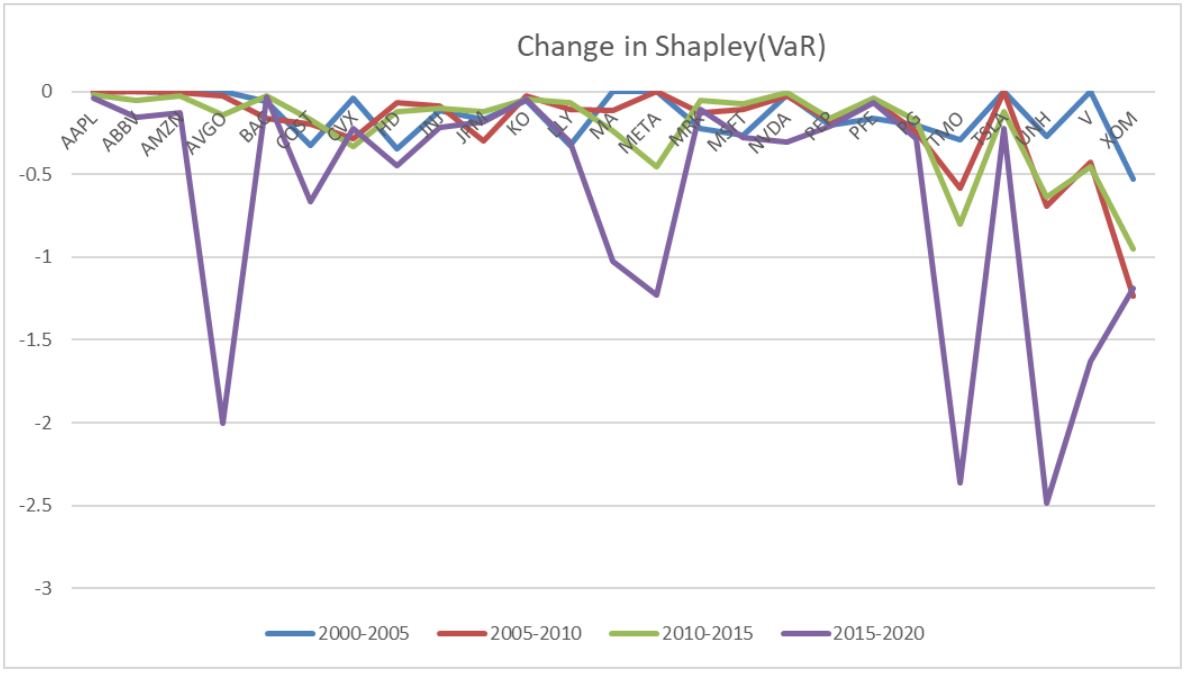
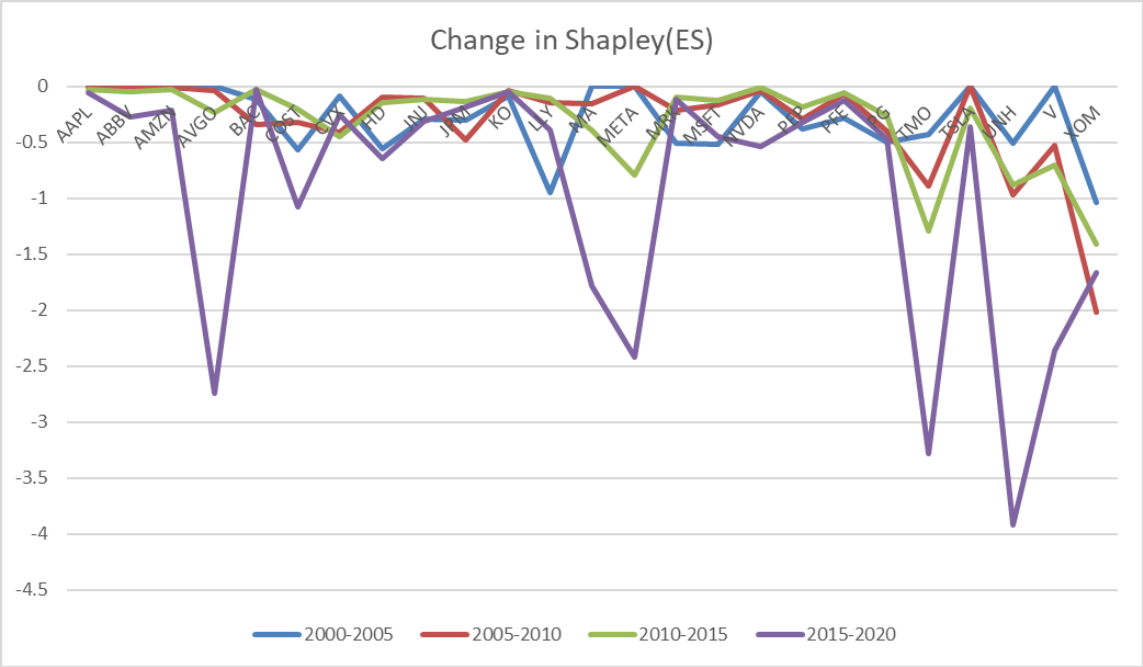


Figure 7. Change in Shapley (ES) for the 25 tickers



From the graphs, we can see that.

- There are several companies which consistently contribute little to nothing to the VaR or ES of the portfolio, these being AAPL, BAC, KO, NVDA, and PFE.
- 2015-2020 was the period with the largest VaR and ES overall, with AVGO, META, TMO, and UNH standing out with exceptional and unexpectedly low values.
- Some companies like TMO, UNH, V, and XOM are consistently some of the worst performing companies.

To see the accuracy of the estimation for the Shapley (VaR) and Shapley (ES), we decided to use spearman correlation from the scipy library of Python3, using the 2015-2020 results. The results for the Shapley (VaR) and Shapley (ES) calculations are

SpearmanrResult(correlation=0.8608, p-value=3.3884e-08)

SpearmanrResult(correlation=0.9692, p-value=1.6849e-15)

As can be observed, the correlation for the ES is very high, while the correlation for VaR is still above 85%. The correlation overall is much lower when compared to theoretical data, but that is to be expected since we figured out the closest distribution for each stock rather than have data which equated to an exact distribution.

Hang Seng Index

We decided to do the same experiment for the Hang Seng Index, to see if there will be any major changes. So, we decided to take the daily data from 2015-2020 of 25 of the top 30 largest market cap companies. Below are our results.

Table 8. Hang Seng Index results for Shapley Value Attribution

Company	Shapley_VaR	Estimated_VaR	Shapley_ES	Estimated_ES
TENCENT	-0.0508	-0.0684	-0.0669	-0.0787
BABA-SW	-0.4281	-0.4457	-0.4672	-0.3360
CHINA MOBILE	-0.2223	-0.0779	-0.3254	-0.2411
CCB	-0.2092	-0.0629	-0.2346	-0.1274
HSBC HOLDINGS	-0.0335	-0.0597	-0.0201	-0.0104
AIA	-0.0036	-0.0049	0.0054	0.0050
MEITUAN-W	-1.8279	-2.4142	-2.9537	-2.7748
CNOOC	-0.0440	-0.0787	-0.0463	-0.0637
HKEX	-5.7282	-6.3878	-8.2254	-6.7307
PING AN	0.0097	0.0156	0.0023	0.0021
ICBC	-0.0007	-0.0007	0.0528	0.0371
SHK PPT	-0.2064	-0.0859	-0.2638	-0.2173
BUD APAC	-0.0738	-0.0751	-0.0851	-0.0542
ANTA SPORTS	-0.4499	-0.5572	-0.6678	-0.7634
XIAOMI-W	-0.2778	-0.1636	-0.3045	-0.4035
CITIC	0.0194	0.0159	0.0160	0.0224
CHINA RES LAND	-0.0210	-0.0186	-0.1013	-0.1259
BOC HONG KONG	-0.1406	-0.2204	-0.1648	-0.1711

BANK OF CHINA	-0.1959	-0.1685	-0.2347	-0.1481
BYD COMPANY	-0.3261	-0.4602	-0.5011	-0.5176
MTR CORPORATION	-0.5093	-0.7984	-0.5853	-0.7529
GALAXY ENT	-0.1069	-0.0946	-0.2265	-0.3412
SANDS CHINA LTD	-1.3326	-0.6092	-2.2507	-1.4699
CHINA OVERSEAS	0.2164	0.1274	0.1598	0.1651
HANG SENG BANK	-5.6000	-5.1394	-5.6000	-4.5207

As can be observed, the magnitude of losses is larger here when compared to S&P 500 in 2015-2020. There are several companies that have very low negative or even positive values, but the negatives among the Hang Seng index are even lower.

Conclusion

The focus our project was to use Shapley Value for Risk and Return Attribution. During our research we realized that previous papers were very heavily focus on proving their experiments while using generated data, thus limiting their application in any real-world problems. Thus, one of the focuses of our paper was to implement the experiments with real data, observe the changes and difficulties in doing so and figure out how to resolve them, not specific to the data itself but as a general solution.

Through this paper we have showcased that Shapley Value and its utilization for Portfolio Assessment, and Risk and Return Attribution can be implemented with some key changes to real world data. Thus, expanding the applicability of Shapley Value to real world problems. We also compared Shapley value with other attribution methods and proved its advantages over others in different scenarios.

Future Work: One of our main limitations was a lack of computation power, therefore when we were doing Monte Carlo Simulations for calculating Shapley Value, we only used 500 iterations, Hagan et al. (2021) used 100,000 iterations. Therefore, one of the ways to improve on our current results would be to redo the simulations with more iterations. Another limitation of conducting the trading simulations was the complexity of implementing convex optimization problems with more factors like momentum, value, growth, commonly used in style investing. It would give more insights into the impact of common trading strategies on the portfolio performance. Another future development might include more performance metrics to the performance attribution.

References

Ancona, M., Ceolini, E., Oztireli, C., and Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In 6th International Conference on Learning Representations (ICLR 2018), 2018.

Ancona, M., Öztireli, C., & Gross, M. (2019). Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Values Approximation. International Conference on Machine Learning, 272–281.

Bacon C. R. Practical portfolio performance measurement and attribution. Vol. 546. John Wiley & Sons, 2008.

Boyd S., Busseti E., S. Diamond, R. N. Kahn, K. Koh, P. Nystrup, J. Speth. Multi-Period Trading via Convex Optimization. Foundations and Trends in Optimization, vol. 3, no. 1, pp. 1–76, 2016.

Colini-Baldeschi, R., Scarsini, M., and Vaccari S. Variance allocation and Shapley value. Methodology and Computing in Applied Probability 20.3 (2018), pp. 919–933.

Garratt, R. J. (2021). An application of Shapley Value Cost Allocation to liquidity savings mechanisms. Journal of Money, Credit and Banking, 54(6), 1875–1888. https://doi.org/10.1111/jmcb.12853.

Hagan P. S., Lesniewski A., Skoufis G. E., and Woodward D. E. (2021). Portfolio risk allocation through Shapley value. URL: https://arxiv.org/abs/2103.05453

Kevin Leyton-Brown. (n.d.). Retrieved December 17, 2022, from https://www.cs.ubc.ca/~kevinlb/teaching/cs532l%20-%202007-8/lectures/lect23.pdf

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems (Vol. 30).

H. Markowitz. Portfolio selection. Journal of Finance, 7(1):77–91, 1952.

Moehle, N. and Boyd, S. and Ang, A. (2021). Portfolio Performance Attribution via Shapley Value. URL: https://arxiv.org/abs/2102.05799

Moulin, H. Fair division and collective welfare. MIT press, 2004.

Shapley. L. S. (1953). A value for n-person games. Contributions to the Theory of Games 2.28 , pp. 307–317.

Štrumbelj, E., Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. Knowledge and information systems 41.3: 647- 665.

Yahoo Finance API. Available at: https://github.com/ranaroussi/yfinance

Appendix

Appendix A. Shapley Value Attribution for VaR and ES

A1. Python3 Code for Shapley Value Calculation through Shapley Value

```
def calc_shapley_vals(char_fct, num_players: int, sample_size: int, VaR: bool) -> np.array:
    """
    char_fct - characteristic function
    num_players - a number of players
    """
    N = set(np.arange(0, num_players))
    shapley_vals = np.zeros(num_players)
    for i in range(num_players):
        N_minus_i = list(N - {i})
        shap = 0.0
        for j in range(sample_size):
            k = np.random.randint(0, num_players - i)
            S = np.random.choice(N_minus_i, k, replace=False)
            S_union_i = np.append(S, i)
            shap += char_fct(S_union_i, VaR) - char_fct(S, VaR)
        shapley_vals[i] = shap / sample_size
    return shapley_vals
```

A2. Python3 Code for character function in Shapley Value calculation

```
def char_func(chosen_elements: np.array, VaR: bool):
    if len(chosen_elements) != 0:
        current_cols = data[:, chosen_elements]
        returns = [y - x for x, y in zip(current_cols, current_cols[1:])]
        minimum = []
        for row in returns:
            a = row.tolist()
            a = [i for i in a if i != 0]
            minimum.append(min(a, default=999))
        minimum = [i for i in minimum if i != 999]
        if len(minimum) == 0:
            return 0
        else:
            value = np.floor(len(minimum) * 0.05 - 1).astype(int)
            minimum.sort()
            if VaR:
                variable = minimum[value]
            else:
                if value == 0:
                    variable = minimum[value]
                else:
                    temp = minimum[:value]
                    variable = sum(temp) / value
            return variable
    else:
        return 0
```

A3. Name for Company Ticker

'AAPL': Apple (Tech, Hardware, Software)

'ABBV': Abbvie Inc. (Healthcare)/(Pharmaceutical)

'AMZN': Amazon (Tech, Service Provider)

'AVGO': Broadcom Inc. (Tech - Semiconductor)

'BAC': Bank of America (Banking)

'COST': Costco Wholesale (Retail)

'CVX': Chevron Corporation (Oil)

'HD': Home Depot (Retail)

'JNJ': Johnson and Johnson (Healthcare)/(Pharmaceutical)

'JPM': JPMorgan Chase & Co. (Banking)

'KO': Coca-Cola Co. (Beverage)

'LLY': Eli & Lilly Co. (Healthcare)/(Pharmaceutical)

'MA': Mastercard Inc. (Payment Processing)/(Financial Services)

'META': META/ Facebook (Tech)

'MRK': Merk and Co. Inc. (Healthcare)/(Pharmaceutical)

'MSFT': Microsoft (Tech, Hardware, Software)

'NVDA': NVIDIA (Tech - Semiconductor)

'PEP': PepsiCo, Inc. (Food, Beverage)

'PFE': Pfizer Inc. (Pharmaceutical)/(Biomedical)

'PG': Procter and Gamble (Manufacturing and Marketing)

'TMO': Thermo Fisher Scientific Inc. (Pharmaceutical)/(Biomedical)

'TSLA': Tesla (Tech)

'UNH': United Health Group (Healthcare)/(Pharmaceutical)

'V': Visa Inc. (Payment Processing)/(Financial Services)

'XOM': Exxon Mobile Corp (Oil)

A4. Company Summary Statistics

Ticker	mean	median	variance
AAPL	13.6431	6.149027	240.2112
ABBV	50.99375	46.91159	271.1421
AMZN	18.04874	6.0035	670.3609
AVGO	98.52008	67.76908	6906.777

BAC	19.8428	18.39059	82.11153
COST	77.94561	47.11916	3959.442
CVX	53.3154	48.68353	766.6224
HD	58.11879	28.44855	2751.425
JNJ	57.98033	42.90413	963.9845
JPM	39.46897	29.48583	605.6267
KO	22.94106	18.54004	107.1989
LLY	45.62528	36.3587	530.6217
MA	74.69655	49.27923	4947.947
META	109.5879	109.89	3236.576
MRK	31.62128	26.91824	198.9386
MSFT	34.96939	21.72524	842.0622
NVDA	10.18788	3.533893	255.5899
PEP	53.45057	44.71516	720.6232
PFE	17.70822	15.70298	48.30084
PG	47.85792	42.97717	457.0019
TMO	79.90146	49.95012	5182.484
TSLA	12.21023	14.10633	58.25946
UNH	68.33599	41.31557	4662.014
V	60.578	49.20689	2167.738
XOM	44.28061	47.13566	270.5629

A5. Simulation Results

Ticker	Shapley_VaR	Estimated_VaR	Shapley_ES	Estimated_ES
AAPL	-0.0015	-0.0007	-0.0031	-0.0039
ABBV	0.0000	0.0000	0.0000	0.0000
AMZN	-0.0054	-0.0087	-0.0135	-0.0092
AVGO	0.0000	0.0000	0.0000	0.0000
BAC	-0.0624	-0.0552	-0.1130	-0.0862
COST	-0.3233	-0.3579	-0.5653	-0.6182
CVX	-0.0430	-0.0568	-0.0849	-0.1120
HD	-0.3432	-0.3714	-0.5529	-0.2545
JNJ	-0.1162	-0.1725	-0.2944	-0.3219
JPM	-0.1681	-0.1031	-0.2994	-0.3918
KO	-0.0513	-0.0095	-0.0880	-0.0468
LLY	-0.3237	-0.4648	-0.9533	-0.5333
MA	0.0000	0.0000	0.0000	0.0000
META	0.0000	0.0000	0.0000	0.0000
MRK	-0.2211	-0.1120	-0.5124	-0.5096
MSFT	-0.2659	-0.3533	-0.5180	-0.7690
NVDA	-0.0215	-0.0280	-0.0429	-0.0335
PEP	-0.2001	-0.2149	-0.3817	-0.4868
PFE	-0.1612	-0.1347	-0.2782	-0.2016
PG	-0.2035	-0.1750	-0.4979	-0.4568
TMO	-0.2917	-0.2102	-0.4306	-0.6084
TSLA	0.0000	0.0000	0.0000	0.0000
UNH	-0.2706	-0.4064	-0.5048	-0.4825
V	0.0000	0.0000	0.0000	0.0000
XOM	-0.5297	-0.4077	-1.0337	-1.2443

Table 4. Empirical results for Shapley Attribution from 2000-2005

Ticker	Shapley_VaR	Estimated_VaR	Shapley_ES	Estimated_ES
AAPL	-0.0064	-0.0116	-0.0087	-0.0099

ABBV	0.0000	0.0000	0.0000	0.0000
AMZN	-0.0086	-0.0131	-0.0124	-0.0168
AVGO	-0.0289	-0.0172	-0.0382	-0.0420
BAC	-0.1628	-0.2472	-0.3420	-0.2873
COST	-0.1973	-0.1713	-0.3225	-0.4148
CVX	-0.2832	-0.4462	-0.4097	-0.4165
HD	-0.0645	-0.0895	-0.0929	-0.0950
JNJ	-0.0901	-0.1427	-0.1028	-0.0598
JPM	-0.2994	-0.2083	-0.4752	-0.2972
KO	-0.0273	-0.0477	-0.0341	-0.0445
LLY	-0.1095	-0.0483	-0.1419	-0.1880
MA	-0.1121	-0.1380	-0.1593	-0.1286
META	0.0000	0.0000	0.0000	0.0000
MRK	-0.1289	-0.1565	-0.2189	-0.1333
MSFT	-0.1083	-0.0406	-0.1643	-0.0784
NVDA	-0.0293	-0.0423	-0.0425	-0.0540
PEP	-0.1725	-0.1739	-0.2930	-0.3107
PFE	-0.0506	-0.0897	-0.0718	-0.0509
PG	-0.2447	-0.3313	-0.4011	-0.5778
TMO	-0.5864	-0.3404	-0.8862	-0.8569
TSLA	0.0000	0.0000	0.0000	0.0000
UNH	-0.6917	-0.9647	-0.9668	-0.6261
V	-0.4282	-0.6536	-0.5301	-0.2732
XOM	-1.2329	-0.3203	-2.0190	-3.0752

Table 5. Empirical results for Shapley Attribution from 2005-2010

Ticker	Shapley_VaR	Estimated_VaR	Shapley_ES	Estimated_ES
AAPL	-0.0191	-0.0241	-0.0299	-0.0389
ABBV	-0.0505	-0.0468	-0.0518	-0.0623
AMZN	-0.0280	-0.0154	-0.0317	-0.0357

AVGO	-0.1443	-0.1028	-0.2297	-0.1639
BAC	-0.0241	-0.0348	-0.0241	-0.0125
COST	-0.1689	-0.2091	-0.2061	-0.1437
CVX	-0.3332	-0.6017	-0.4466	-0.5598
HD	-0.1223	-0.0720	-0.1467	-0.2121
JNJ	-0.0994	-0.0569	-0.1207	-0.0575
JPM	-0.1192	-0.0398	-0.1319	-0.1922
KO	-0.0441	-0.0126	-0.0444	-0.0585
LLY	-0.0655	-0.0429	-0.1072	-0.1479
MA	-0.2344	-0.1997	-0.3951	-0.4125
META	-0.4575	-0.6043	-0.7899	-0.5898
MRK	-0.0558	-0.0764	-0.0919	-0.0458
MSFT	-0.0710	-0.1224	-0.1260	-0.1003
NVDA	-0.0068	-0.0051	-0.0133	-0.0109
PEP	-0.1587	-0.1313	-0.1861	-0.0980
PFE	-0.0385	-0.0071	-0.0587	-0.0293
PG	-0.1762	-0.2748	-0.2628	-0.1268
TMO	-0.8020	-0.6041	-1.2961	-1.3888
TSLA	-0.1210	-0.0842	-0.1944	-0.2094
UNH	-0.6346	-0.9582	-0.8778	-0.8004
V	-0.4531	-0.3562	-0.7007	-1.0710
XOM	-0.9533	-0.9611	-1.4119	-0.9065

Table 6. Empirical results for Shapley Attribution from 2010-2015

Appendix B. Code for Simple Return Attribution Experiment

The code is to calculate the one-at-a-time, sequential, shapley attribution, given the data with 2 features. *data* variable is the input data used in experiment 2 of this paper.

```
data = pd.Series({
    (0,0): 7.2,
    (1,0): 5.1,
    (0,1): 9.4,
    (1,1): 8.6,
})

result = pd.DataFrame()

# one-at-a-time attribution
b = data[(0,0)]
a1 = data[(1,0)] - data[(0,0)]
a2 = data[(0,1)] - data[(0,0)]
unattr = data[(1,1)] - data[(1,0)] - data[(0,1)] + data[(0,0)]
result = result.append({"method": "one-at-a-time", "b": b, "a1": a1, "a2": a2, "unattributed": unattr,
}, ignore_index=True)

# Sequential attribution
b = data[(0,0)]
a1 = data[(1,0)] - data[(0,0)]
a2 = data[(1,1)] - data[(1,0)]
unattr = data[(1,1)] - a1 - a2 - b
result = result.append({ "method": "sequential", "b": b, "a1": a1, "a2": a2, "unattributed": round(unattr, 3),
}, ignore_index=True)

# Shapley attribution
b = data[(0,0)]
a1 = 0.5 * (data[(1,1)] - data[(0,1)] + data[(1,0)] - data[(0,0)])
a2 = 0.5 * (data[(1,1)] - data[(1,0)] + data[(0, 1)] - data[(0,0)])
unattr = data[(1,1)] - a1 - a2 - b
result = result.append({"method": "shapley", "b": b,"a1": a1, "a2": a2,"unattributed": round(unattr, 3),
}, ignore_index=True)
```

Appendix C. Code for Portfolio management via Convex Optimization

We attached the code to run convex optimization using the custom class from CVXPortolio library in Python. The code below runs optimization scenario when all hyperparameters are turned on or included.

```
r_hat = returns.rolling(window=250, min_periods=250).mean().shift(1).dropna()
Sigma_hat = returns.rolling(window=250, min_periods=250, closed='neither').cov().dropna().droplevel(1)

tcost_model=cp.TcostModel(half_spread=10E-4)
hcost_model=cp.HcostModel(borrow_costs=1E-4)

risk_model = cp.FullSigma(Sigma_hat)
gamma_risk = 5.
gamma_trade = 1.
gamma_hold = 1.
leverage_limit = cp.LeverageLimit(1)

spo_policy = cp.SinglePeriodOpt(return_forecast=r_hat,
                                costs=[gamma_risk*risk_model,
                                       gamma_trade*tcost_model,
                                       gamma_hold*hcost_model],
                                constraints=[leverage_limit])

market_sim=cp.MarketSimulator(returns, [tcost_model, hcost_model], cash_key='USDOLLAR')
init_portfolio = pd.Series(index=returns.columns, data=250000.)
init_portfolio.USDOLLAR = 0
results = market_sim.run_multiple_backtest(init_portfolio,
                               start_time='2018-01-02',  end_time='2022-12-30',
                               policies=[spo_policy, cp.Hold()], parallel=False)
```