Portfolio Optimization System

Yichao Lin Lin Tan Yujie He

The Fu Foundation School of Engineering and Applied Science, Columbia University

500 W. 120th Street #510, New York, NY 10027

Executive Summary

We establish and expand the functionality and modularization of a fully automated system to optimize the stock selection and asset allocation of an equity-cash portfolio. The system is composed of four essential modules: the Asset Selection Module (ASM), which utilizes deep reinforcement learning to extract insights from historical prices and pick the most promising stocks accordingly; the Position Evaluation Model (PEM), which generates the performance scores for stock and determines the weight of a stock in the portfolio using both deep learning and machine learning techniques with company fundamentals and analyst sentiments as features; the Macroeconomic Evaluation Module (MEM), which leverages dimensionality reduction and support vector machine to predict macro market performance and dictates the allocation of capital between cash and equity; and a Backtesting module, which organizes the information generated by the aforementioned 3 modules to manage asset allocation and dividend reinvestment. Plus, we construct a minimal variance portfolio with constraints as initialization and implement the rule-based trading strategy during backtesting. The system produces a good risk-adjusted portfolio, beating the benchmark curve and yielding satisfying returns.

The purpose and methodology of these modules are informed by the investment theory and industry experience of Howard Hissrich, a wealth manager at Morgan Stanley. Through analyzing his trade logs and portfolio notes, we identify the key heuristics and bring the ideas to feature engineering, stock classification, asset diversification, and portfolio rebalancing. We are confident that the whole system could be applied by a wealth manager to improve performance on the pertinent problem of portfolio composition and optimize the products.

1 Introduction

In light of the previous work and the investment strategy of Howard Hissrich at Morgan Stanley, our group's project is to enhance and update the Portfolio Optimization System (POS) which optimizes the composition and overall performance of a portfolio of high-dividend stocks held by a portfolio manager. In the fall 2019 semester, the last group leverages reinforcement learning, machine learning and deep learning techniques to automate the process of portfolio management, incorporating many of the company fundamentals and macroeconomic factors that a successful wealth manager takes into account.

Our work is a continuation of the project conducted by Kai Wu and Connor Goggins, which demonstrated that a feed-forward neural network with fundamental factors and analyst sentiments as input features did have predictive power for future stock returns. We take this project a step further by analysing the information coefficient and information ratio of features, incorporating techniques such as augmenting the neural network, adding batch normalization and dropout in each layer. Moreover, we utilize the decision tree and random forest algorithm to train the model on a rolling basis, which improves the training and test accuracy significantly.

The remainder of this report is structured as follows: Section 2 reviews the previous work; Section 3 presents the four core modules of the system and the improvement we have made accordingly; Section 4 summarizes the data source used in our project; Section 5 presents the key results; Section 6 concludes the report and brings out the future work suggestions.

2 Previous Work

The previous group designed Portfolio Optimization System (POS) in a modular fashion, with four distinct modules that inform each other and work in tandem towards increasing the total P&L of the portfolio. The Asset Selection Module (ASM) leverages deep reinforcement learning to pick stocks. The Position Evaluation Model (PEM) uses deep neural networks to evaluate whether a position is underweight or overweight so as to trigger position adjustment signals. The Macroeconomic Evaluation Module (MEM) applies economic factors to measure the desirability of equity investing and will thus influence the asset allocation between equities and cash. The backtesting module takes care of the dividend reinvestment framework and is responsible for the rule-based trading strategy and evaluation of the entire POS system. This final module serves to both organize and utilize the information flow generated by the aforementioned three modules.

The insights of these modules are informed by the investing theory and experience of Howard Hissrich, a wealth manager at Morgan Stanley. They analyzed his trade logs and portfolio notes, and identified key heuristics, most of which were related to fundamental factors that were directly tied to his decision to buy, sell, or hold any given stock.

As we worked towards the goal of refining the POS, we drew on Kai Wu's work from the fall semester. The approach and the architecture that he developed in the PEM part proves to be invaluable for this semester's project. We extend beyond the PEM part and generate 3 different sets of new scores based on the gvkey of each stock.

3 Module Structure and Design

In this part, we will elaborate on the architecture of our system and the improvement we have incorporated into each module in detail. Asset Selection Module (ASM) will be used to pick initial stocks from the American equity market at the beginning of the investment horizon. After that, both the Position Evaluation Module (PEM) and the Macroeconomic Evaluation Module (MEM) will be triggered monthly. The PEM will evaluate stocks in which the portfolio currently holds positions and adjusts the weight of stocks based on certain thresholds. Concurrently, the MEM will suggest a target cash-stock ratio. If after trimming overweight positions the cash-stock ratio is higher than the target, the system will increase the underweight positions or trigger the PEM again to long new stocks. Figure 1 gives an overview of the interaction between four modules.

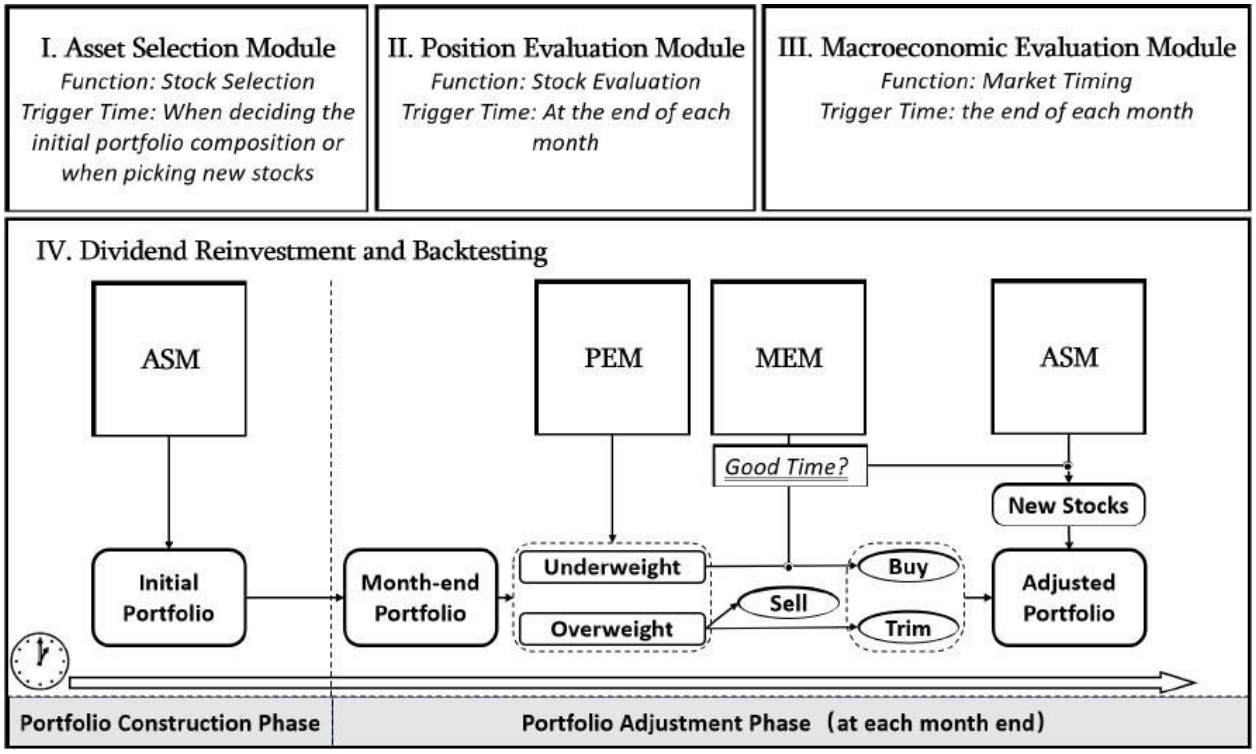


Figure 1: Interactions between Four Modules

3.1 Asset Allocation Module (ASM)

This module uses deep reinforcement learning (Deep Q Networks with Prioritized Experience Replay and Target/Principal Networks) to predict the future value of holding a position in any given stock in the universe of high-dividend American stocks (dividend 3%). The ASM takes stock performance over the past 32 days and macroeconomic signals into account and is utilized to select the initial stocks for the portfolio before the Position Evaluation Module (PEM). The loss function for the deep Q networks is defined according to the gains and losses experienced over the training period, such that the model is rewarded if the stock performs well and is penalized otherwise. As training the deep Q networks, it learns to predict the price movements of stocks in the universe (high-dividend American stocks). Once the model is fully trained, it can be used to generate a Q-score for any stock and select the most promising stocks with a long-term horizon in mind.

3.2 Position Evaluation Module (PEM)

The purpose of PEM will be to generate the performance scores of each stock indicating future price movement for each active stock in the entire stock pool. We construct two major classification models with a ternary classification model and a 21-classes classification model respectively.

Since the absolute returns of individual stocks are continuous and noisy in nature, the neural networks we built are classification models rather than regression models. Based on 5-layer feed-forward neural networks with batch normalization and dropout on each layer, we aim to predict the monthly returns of each stock using the features we selected.
To boost the predictive performance, we combine the prediction results of both the ternary classification model and the 21-classes classification model to produce a synthetic score of each stock in the portfolio. In order to reflect the real market anticipation of each stock and make predictions as accurate as possible, we select 29 features as a mixture of company fundamentals and analyst sentiments. We will demonstrate a detailed feature definition and data analysis steps in Section 4 Data Description.

The PEM will decide whether any of the positions currently held in our portfolio are overweight or underweight once the predictions for stock price movement are ready. The dataset in the ‘PEM_data.csv’ is rectangular, with each row representing one stock observation at one period. Each observation includes the values of various features as well as the return of the subsequent month.

The configuration of two classification models with ternary classification and 21-classes classification is described as follows:

3.2.1 Ternary Classification Model

We divide the stocks into 3 categories based on their absolute monthly returns:

- Class 0: monthly gain of more than +5%
- Class 1: monthly return between -7% and +5%
- Class 2: monthly loss of more than 7%

As an upgrade on the previous group architecture, the neural network architecture we implement is four hidden layers (29×256×128×64×32×3) with corresponding batch normalization and dropout on each layer to avoid overfitting. Plus, we apply Adam optimizer to train the network. We use Relu as the activation function for hidden layers and Softmax for the output layer as before.

3.2.2 21-Classes Classification Model

The 21-classes classification model is much more fine-grained than the ternary classification model. The bigger is the class number, the higher is the future monthly return.

- Class 0: monthly loss of more than -12%
- Class 1: monthly loss between -12% and -8%
- Class 2: monthly loss around -7%
- Class 3: monthly loss around -6%

……

- Class 9: monthly basically no gain or loss
- Class 10: monthly return around +1%
- Class 11: monthly return around +2%

……

- Class 19: monthly return between +10% and +15%
- Class 20: monthly gain of more than +15%

As an improvement on the previous group architecture, the neural network architecture we implement is four hidden layers (29×512×256×128×64×21) with corresponding batch normalization and dropout on each layer to avoid overfitting. In addition, we use the callback function to perform early stopping once our validation accuracy hits a certain threshold. We apply RMSProp optimizer to train the network. The Relu is the activation function for hidden layers and Softmax for the output layer as before.

3.2.3 Decision Tree and Random Forest

Since the accuracy of the neural network built by the last group was lower than 50%, our group aims at raising the accuracy for the classification prediction model. To start with,

we improve the neural network as we mentioned in the last section. Although the prediction accuracy of the updated neural network has increased to 55%, it is still not accurate enough. Therefore, our group explores to build a new prediction model which could significantly increase the prediction accuracy. Firstly, we plot a correlation map of all the features and the stock returns. However, we find that there is no linear correlation between the features and the returns. Therefore, our next step is to find an appropriate non-linear classification model.

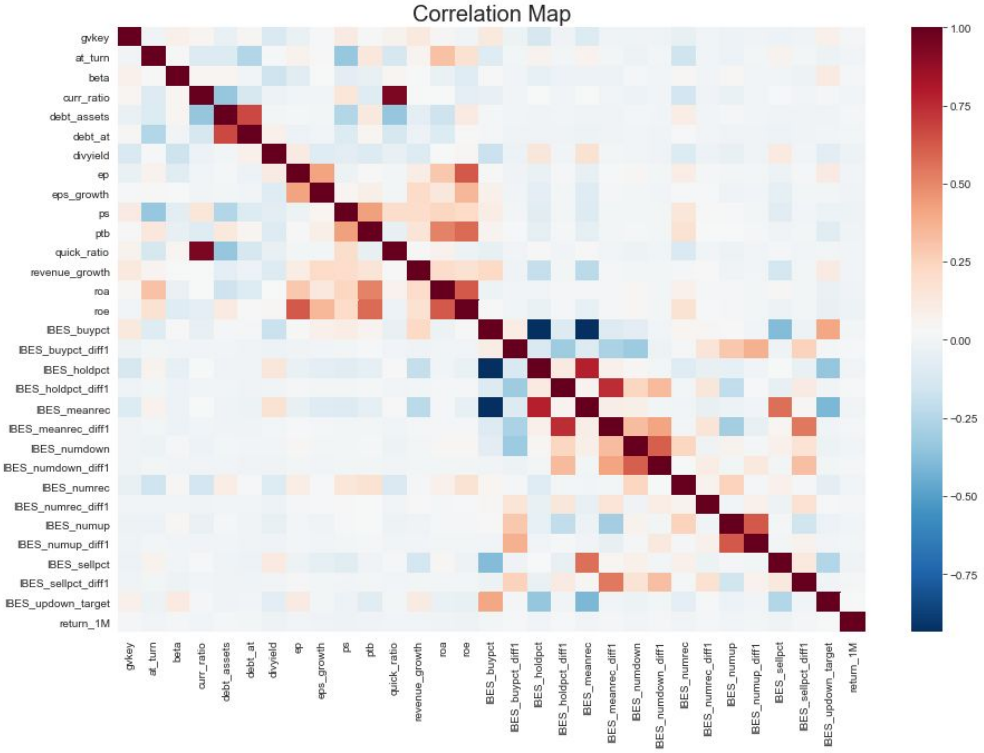


Figure 2: Correlation Map of the PEM features

We include two additional machine learning algorithms to the PEM part since the deep learning algorithm does not provide that much growth on the test set as we expect. Surprisingly, the decision tree and random forest would have a higher prediction accuracy than neural networks as these algorithms can generalize well to test cases and avoid overfitting. We utilize the decision tree algorithm to perform ternary classification given that next month stock's return could be foreseen through one or two significant factors if the factor shows an early signal of the bullish or bearish market. This is where the decision tree comes to play.

Through complex combinations of features to generate various branches, the decision tree could classify these stocks to three separate return levels accordingly. Moreover, we grid search the depth of the tree and use 10-fold cross-validation to decide the optimal tree's depth, which is 10.

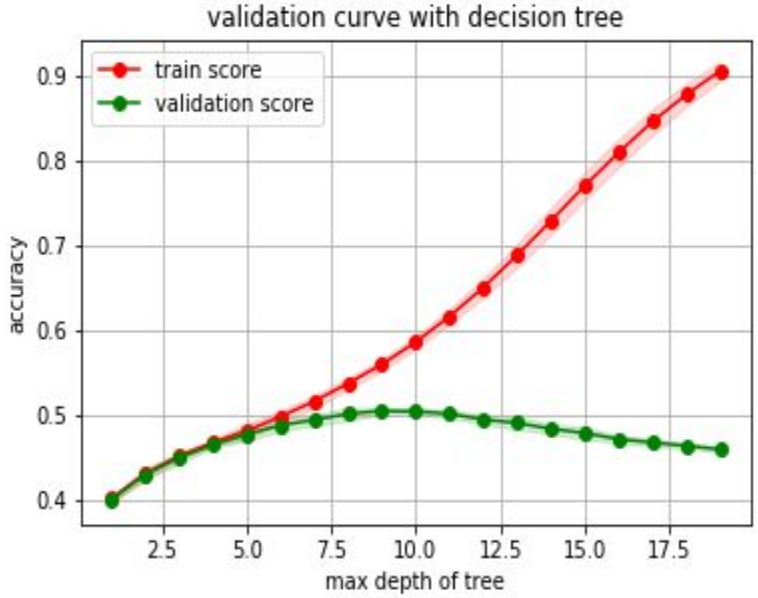


Figure 3: Decision Tree

In terms of 21-Classes classification, considering that there would be many branches of the tree, so it would be more logical if we have multiple trees to decide the return levels. Hence, the random forest is a better choice of training algorithm than decision tree in view of the multiple classification problem.

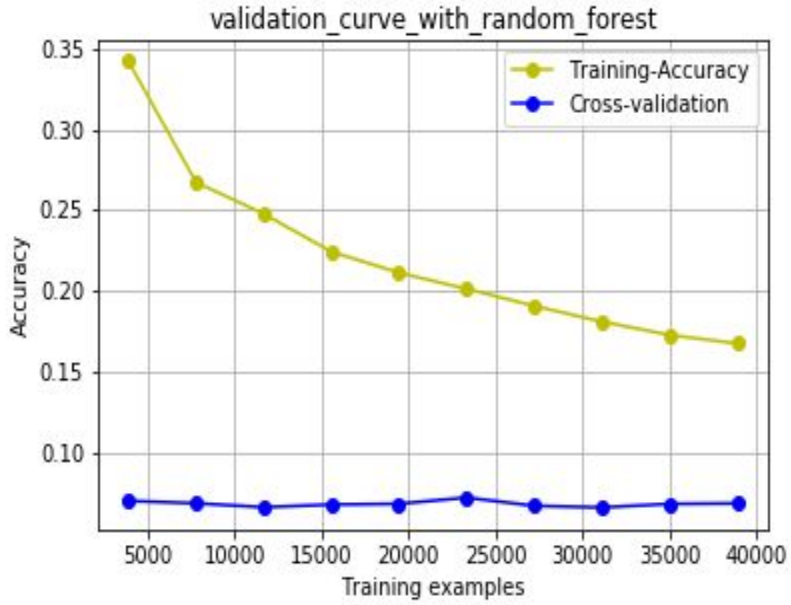


Figure 4: Random Forest

3.2.4 Training and Testing result comparisons

We demonstrate the accuracy comparisons between the previous group and our group by applying well-tuned deep learning and machine learning algorithms as follows:

Ternary Classification Model				
	Last Group	Our Group	Decision Tree	Rolling ML
Train Accuracy	0.4519	0.5510	0.8577	0.8834
Test Accuracy	0.3587	0.3469	0.3576	0.3777
21-Classes Classification Model				
	Last Group	Our Group	Random Forest	Rolling ML
Train Accuracy	0.1101	0.1375	0.1769	0.1921
Test Accuracy	0.0793	0.0616	0.0744	0.0778

Table 1: Accuracy Comparisons among Different Methods

3.3 Macroeconomic Evaluation Module (MEM)

The key purpose of the Macroeconomic Evaluation Module (MEM) is for market timing. Portfolio Optimization System (POS) depends on this module to allocate the stock-cash ratio for the following month: if the market is favorable, then 97% of the available capital will be spent on stocks; otherwise, the ratio reduces to 92% and the rest 8% is put in the fixed-income market. This stock-cash ratio here is consistent with Howard's strategy.

The crux of the MEM is a vanilla binary classification model used to predict the market trend in a future period of time. The label for prediction is set to be Favorable (1, if the market index will rise for more than 1%) or Unfavorable (0, the market index will not rise for more than 1%). During backtesting implementation, we set the horizon to be one month, which is consistent with the frequency of evaluating and adjusting the portfolio from the previous PEM module.

The previous group used machine learning algorithms such as Random Forest and Logistic Regression to train the model. We perform the dimensionality reduction on the input dataset first and then apply Support Vector Machine (SVM) to train the model.

Besides, we use grid search and cross-validation to select the best parameter sets and test them on the separate test cases. The best parameters for the SVM is :{'C': 100, 'gamma': 0.001, kernel='rbf'}.

Specifically, the 35 input features include:

• *Momentum*: percentage change of the S&P500 Index in the last 5 days;
percentage change of the S&P500 Index in the last 20 days;

• *Volatility*: absolute level of VIX Index;
percentage change of VIX Index in the last 5 days;
percentage change of VIX Index in the last 20 days;
• *Interest Rate Curve*: 1/3/6 Month, 1/2/3/5/7/10/20 Year Government Bond Rate;
and respective percentage change in the last 5 days and 20 days.

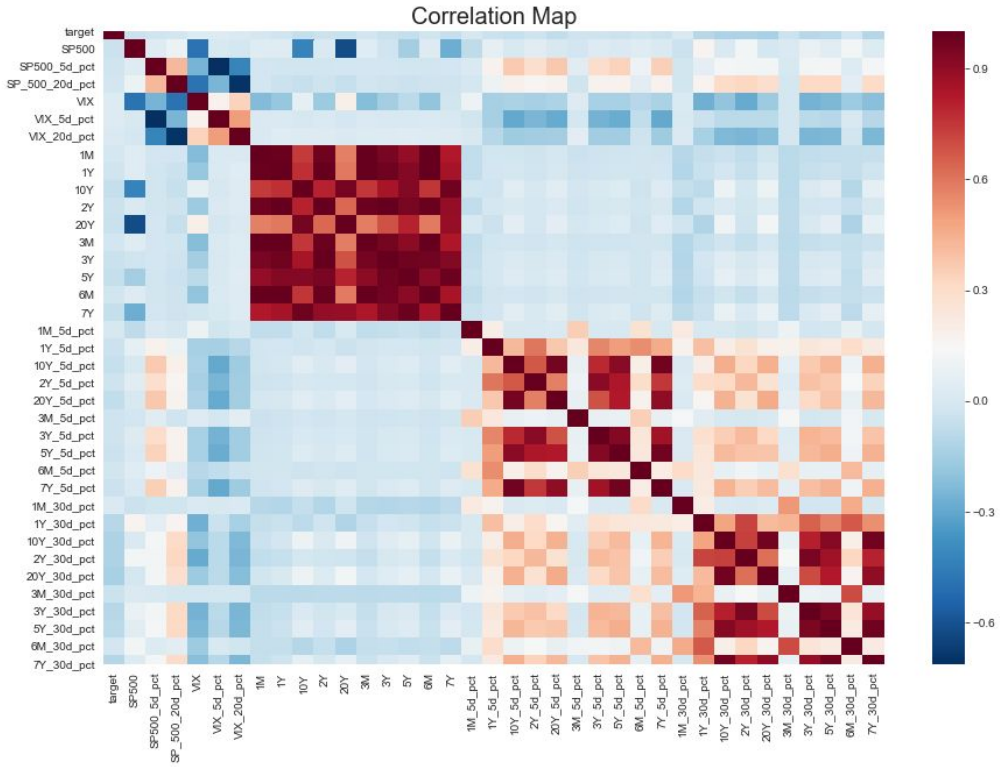


Figure 5: Correlation Map of the MEM features

3.3.1 Dimensionality Reduction

For multidimensional data, we want to perform feature extraction in order to transform the data from the high-dimensional space to a space of fewer dimensions. Then, we can plot and visualize the distribution of different classes. In this MEM part, we apply the principal component analysis (PCA), which makes use of the Singular Value Decomposition (SVD) method and transforms the data into a lower-dimensional space linearly. Moreover, AutoEncoder (AE) employs a neural network with encoder layers and decoder layers and learns non-linear data mapping and codings together with an inverse function from the coding to the original representation. Our AE network architecture is (35×64×32×16×2×16×32×64×35) and we apply Adam optimizer to train the network. The Relu is the activation function for hidden layers and Tanh for the output layer. The key of dimensionality reduction is to visualize the distribution of the training dataset.

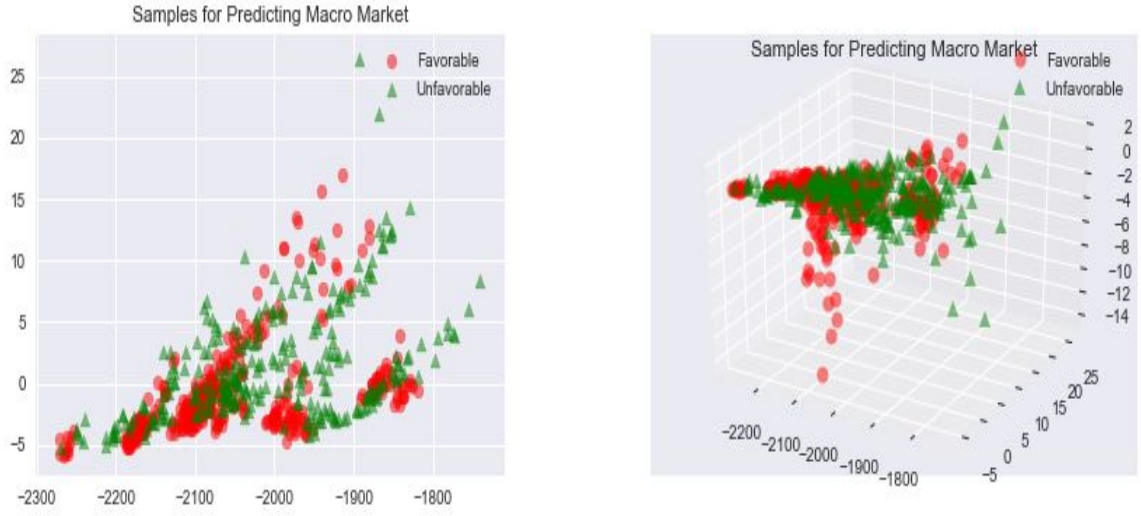


Figure 6: PCA for 2-dimension and 3-dimension data

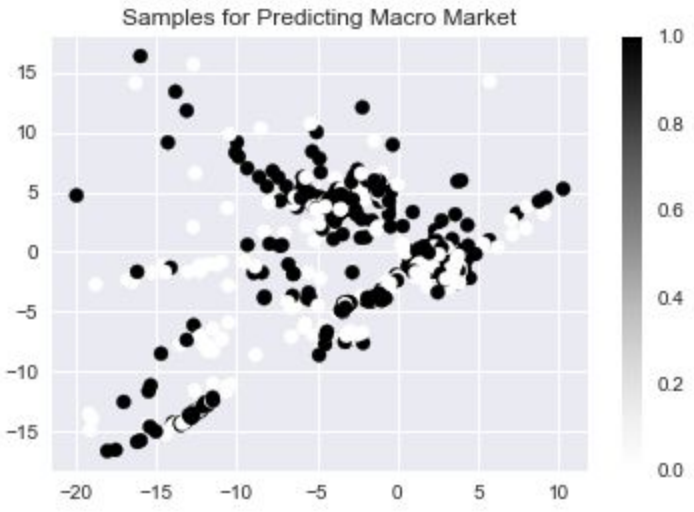


Figure 7: AutoEncoder (AE) for 2-dimension data

3.3.2 Training and Testing result comparisons

We compare our improved models and results with the previous group. Obviously, the dimensionality reduction technique could enhance prediction accuracy considerably.

	Last Group - Logistic Regression	Our Group - PCA + SVM	Our Group - AE + SVM
Training Accuracy	0.7059	0.8211	0.7467
Test Accuracy	0.6849	0.7455	0.6498

Table 2: Accuracy Comparisons among Different Methods

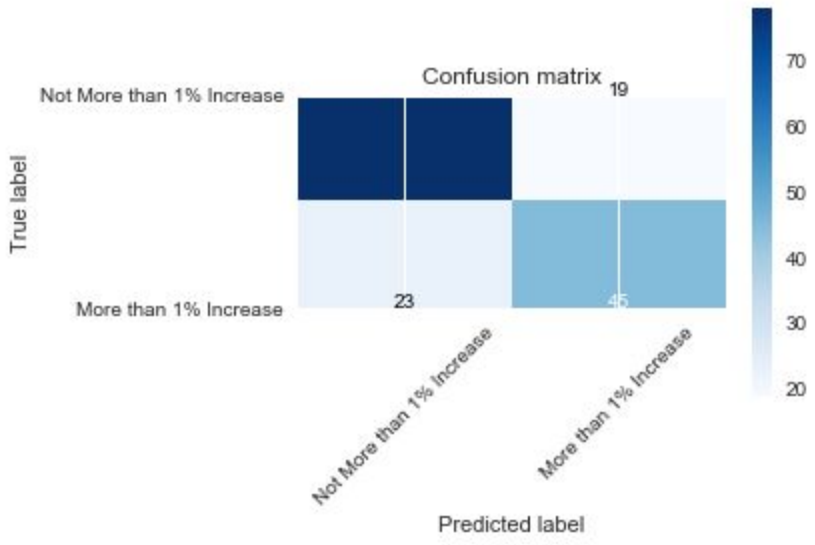


Figure 8: Confusion Matrix

3.4 Dividend Reinvestment and Backtesting Framework

Notably, dividend reinvestment is an indispensable part of nearly every real-world stock portfolio management. During the course of this project, we focus on the high-dividend stocks rather than index constituents. The backtesting module aims to incorporate the portfolio management logic, including stock selection and asset allocation, dividend reinvestment, and market timing. Since the processes involved in receiving and reinvesting dividends are closely tied to the backtest, we integrate the dividend reinvestment policy into the backtesting framework.

3.4.1 Dividend/Cash reinvestment Strategy

Generally, the stocks we select into the stock pool are high-dividend stocks. Throughout the project, we focus on the high-dividend stocks rather than index constituents. Therefore, it is critically essential to make the best use of their dividends. Cash dividends received will increase the cash in the portfolio. Therefore, the question of how to reinvest the dividends is the same as how to deploy the cash.

In light of portfolio diversification and risk management, the amount of cash that the system uses to invest in stocks will depend on the forecast of macroeconomic status generated by the MEM module. If the macroeconomic status tends to be favorable to equity investment, the target amount of cash will decline and the portfolio will weigh more into equities; otherwise, it will spend more money on fixed-income investment like 2-year US Treasury Notes.

We will illustrate the stock investment and backtesting logic of Portfolio Optimization System (POS) as follows:

3.4.2 Backtesting Framework - Portfolio Construction

At the beginning of backtesting, we choose the top 34 stocks based on the PEM scores. In the equity selection part, we primarily focus more on return maximization. Hence, in the asset allocation part, we need to take care of the risk factor of our portfolio by applying the minimum variance asset allocation.

In the previous work, all top 34 stocks in the initial portfolio are equally weighted with around 2.94%, which is an intuitive way when we have no clues on how to decide the structure of the portfolio. However, based on the historical performance of candidate stocks and their accordingly scores, we are able to assign higher weights to those more promising stocks. In order to improve the performance of the portfolio, we introduce the mean-variance model which assists in determining the weights of assets in a portfolio by considering expected returns and the volatility and their relationship. In the context of Markowitz theory, an optimal set of weights is one in which the portfolio achieves an acceptable baseline expected rate of return with minimal volatility. In our case, we don't set a target expected return. Instead, we try to minimize the volatility of returns of constructed portfolios. Thus, the goal is to minimize the risk of the portfolio so that investors can obtain more stable returns. In the meanwhile, we add one more constraint on weights to the original mean-variance model for the purpose of being more consistent with previous work and avoiding bias to one or two assets.

Assume there are n assets, indexed by i and j. Let r_i denote the expected rate of return for asset i, and σ_{ij} denote the covariance of the rate of returns of assets i and j, for i, j =1,2, ..., n. We define the return vector

$$r = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n-1} \\ r_n \end{pmatrix}$$

And the covariance matrix

$$\Gamma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{pmatrix}$$

If w is a set of weights associated with a portfolio, we have the weight vector

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n-1} \\ w_n \end{pmatrix}$$

Then the expected rate of return of this portfolio is

$$\bar{r} = r^T w$$

And the variance of this portfolio is

$$var = w^T \Gamma w$$

Therefore, the optimal weights that minimize the volatility of returns of the constructed portfolio is the solution to the following mathematical program:

$$\begin{aligned} & w^T \Gamma w \ ; \\ s.t \quad & e^T w = 1 \\ & 0 \leq w \leq 0.1; \end{aligned}$$

Where e is an n-dimensional vector of all 1's. The purpose of the last constraint is to make sure that the weights of all assets are bounded between 0.0 and 0.10. Therefore, the portfolio would not place too much weight on a small set of assets. Instead, weights would be more evenly spread on all assets, which is comparable with the equally-weighted case.

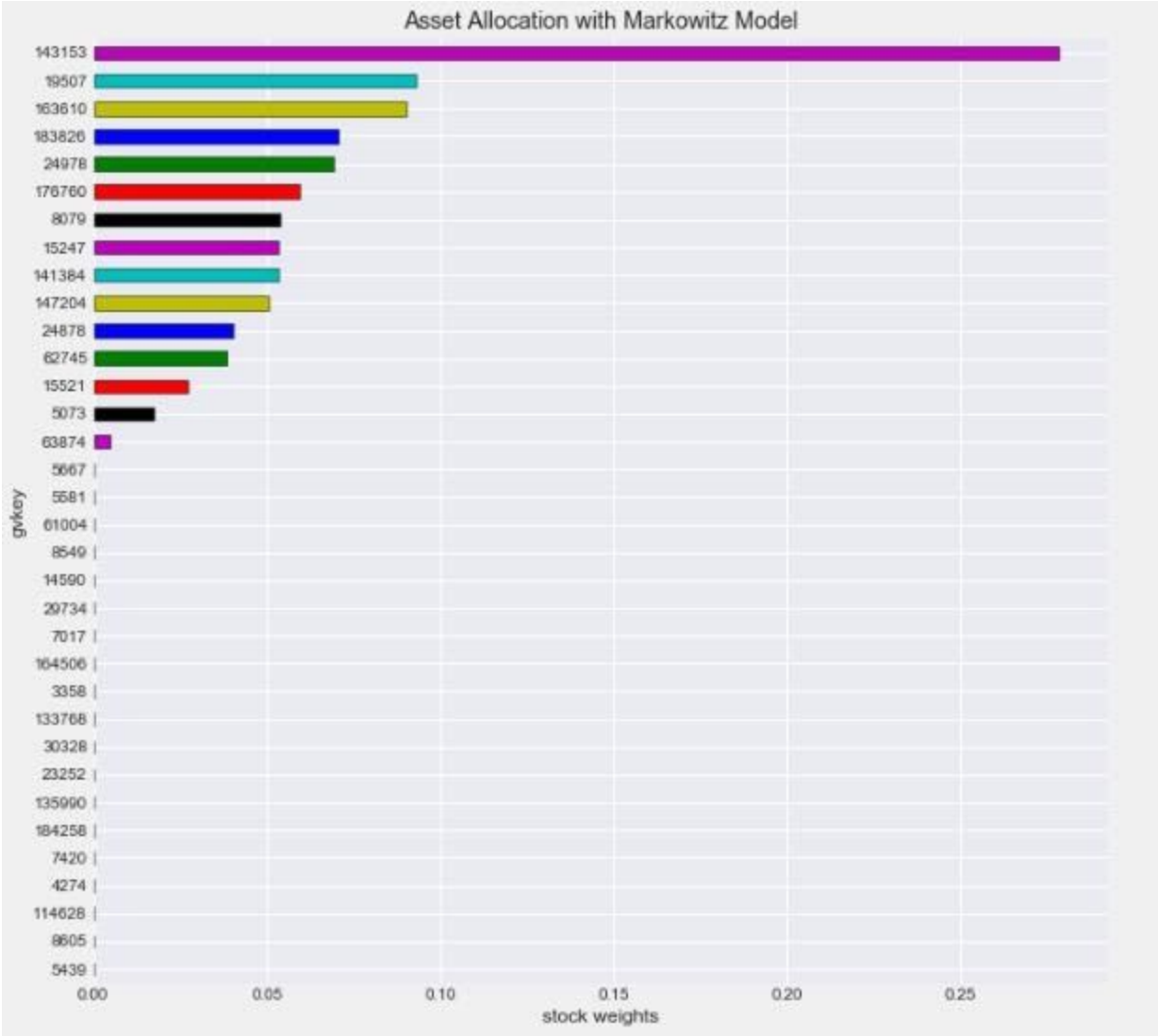


Figure 9: Markowitz allocation with no weight constraints

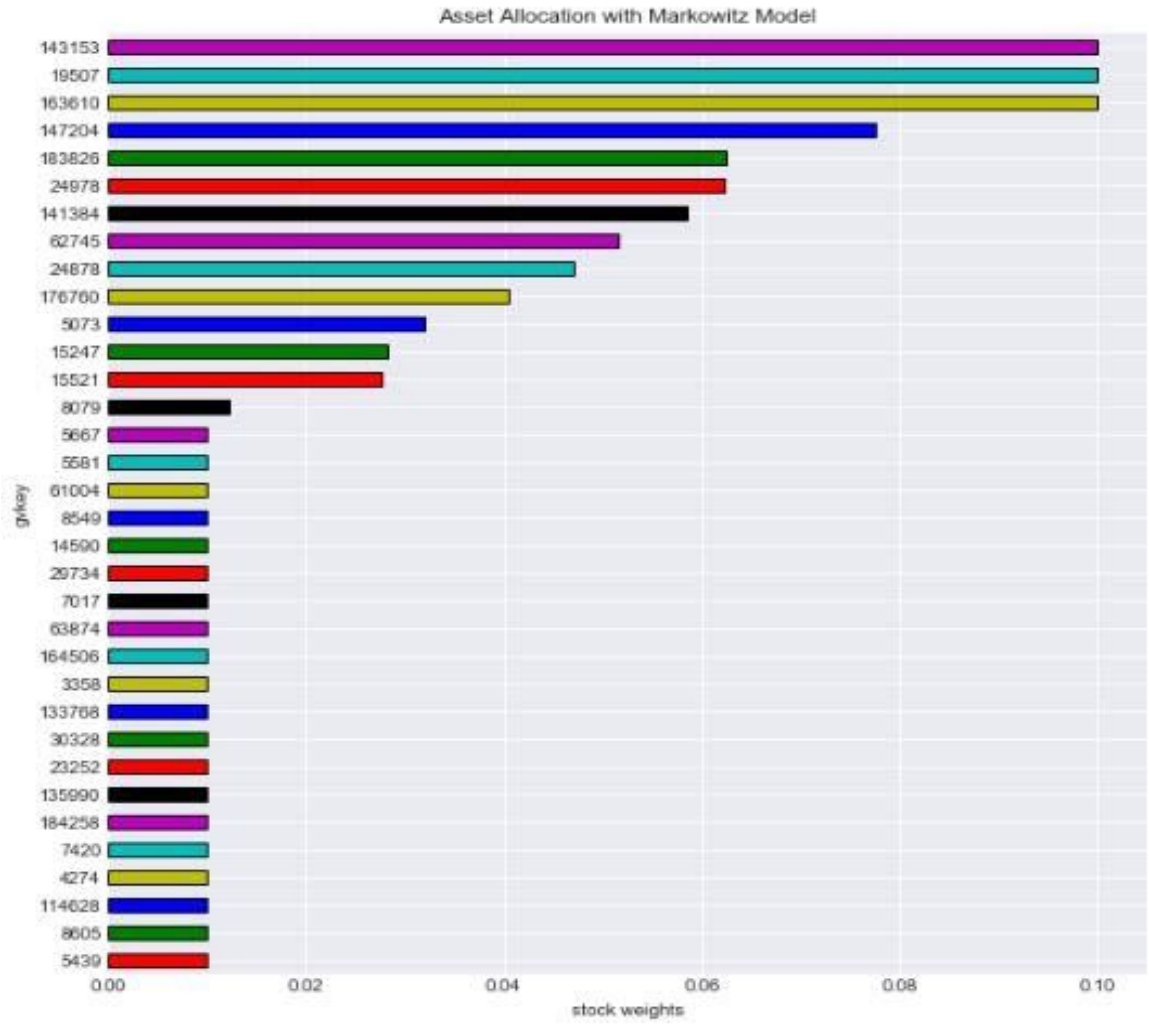


Figure 10: Markowitz allocation with (0.01%, 10.0%) weight constraints

3.4.3 Backtesting Framework - Portfolio Adjustment

We will trigger our rule-based trading strategy at the end of each month, which is a standard way for us to construct a risk-adjusted portfolio.

1. Check stock dividend information in the month
2. Update cash/stock dividends received into the portfolio
3. Use PEM to generate scores and evaluate each stock
4. Use MEM to assess the macro market environment
5. Portfolio Adjustment

   5.1. Keep stocks that are neither overweight nor underweight.

   5.2. For an overweight stock, if its current weight is greater than 2%, trim the weight to 2%; if its current weight is smaller than 2%, sell the stock.

   5.3. If there is cash available, which exceeds the target cash amount suggested by MEM, buy underweight stocks ordered by PEM scores.

   5.4. If the current weight of an underweight stock is greater than 4%, skip it; otherwise buy the stock until it weighs 4% of the total stock portfolio.

5.5. If there is still cash remaining, buy more top-ranked stocks based on PEM scores, so that each newly-bought stock has a weight of 2.94% in the portfolio.

4 Data Description

4.1 Data Overview

The Portfolio Optimization System relies on four categories of financial data:

• *Market Data*: drawn from the CRSP Database; including price, volume, beta, and dividend data for each stock on a daily basis, as well as historic prices of the S&P 500 index; used in all four modules.

• *Company Fundamentals*: drawn from the S&P Compustat Database and WRDS Analytics; including accounting information and other company characteristics like GIC industry category; updated quarterly; used in PEM mostly.

• *Analyst Sentiments*: monthly data drawn from the I/B/E/S Database; including financial analysts' consensus on stock ratings and diverse sources of opinions (long, short, hold) within an interval; used in PEM.

• *Macro-economic Data*: daily data drawn from the Factset Database; including interest rates data and volatility index data; used in MEM.

4.2 Features Data Analysis used in PEM

In the PEM part, we extract 29 features in total as input, most of which are company fundamentals and analyst sentiments. The selection of these features is roughly based on the heuristics extracted from Howard's trading logs.

Table 3 summarizes their variable names in the dataset, definitions, categories, Information Coefficient (IC), and Information Ratio (IR). Noted that the table only has 22 rows. The remaining seven features are the 1st-order difference of IBES_buypct, IBES_holdpct, IBES_sellpct, IBES_meanrec, IBES_numdown, IBES_numup, or IBES_numrec. The variation of these variables could bring more information than the original value, thus we include them as part of our feature sets.

The information coefficient (IC) represents the cross-sectional correlation coefficient of the selected stock's factor value and the stock's next-period return. The IC value can be used to judge the predictive ability of the factor value to the next period's rate of return. The larger the IC value, the larger the factor Alpha. When the IC is greater than 5% or less than -5%, the factor is considered to be more effective. If the average value of IC can reach more than 10%, it means that this factor owns very strong predictive power.

The information coefficient (IR) is equal to the mean of IC over multiple periods divided by IC standard deviation, which represents the ability of the factor to obtain Alpha. Any factor experiences through historically unstable times, so the effectiveness of factors has non-periodic fluctuations in history. The higher the value of IR, the more stable the prediction of the factor. When IR is greater than 50%, the factor has a strong ability to obtain excess returns steadily.

Feature Name	Definition	Category	IC_Mean	IR
beta	the sensitivity of stock return to market return	technical indicator	0.01444	0.06739
at_turn	asset turnover ratio	operating capacity	0.00111	0.01643
divyield	expected dividend yield	dividend	-0.02248	-0.29174
curr_ratio	current ratio	debt capacity	-0.00580	-0.10309
quick_ratio	quick ratio	debt capacity	-0.00411	-0.07723
debt_assets	debt to asset ratio	debt capacity	0.19150	0.09161
debt_at	interest-bearing liability ratio	debt capacity	0.00607	0.09161
eps_growth	annual growth rate of earnings per share	growth potential	-0.00471	-0.08283
revenue_growth	annual growth rate of revenues	growth potential	-0.00051	-0.00932
ep	earnings to price ratio	price multiples	-0.00023	-0.00306
ps	price to sales ratio	price multiples	-0.01620	-0.18998
ptb	price to book ratio	price multiples	-0.00503	-0.05926
roa	return of asset	profitability	-0.00787	-0.11002
roe	return of equity	profitability	-0.00988	-0.11198
IBES_buypct	percentage of "buy" recommend	analyst sentiments	0.00915	0.12476
IBES_holdpct	percentage of "hold" recommend	analyst sentiments	-0.00598	-0.08483

IBES_sellpct	percentage of "sell" recommend	analyst sentiments	-0.00960	-0.16010
IBES_meanrec	average analyst ratings (1 to 5)	analyst sentiments	-0.00910	-0.12325
IBES_numdown	number of downward rating adjustments	analyst sentiments	-0.00066	-0.01306
IBES_numup	number of upward rating adjustments	analyst sentiments	0.00244	0.04589
IBES_numrec	number of analyst reports	analyst sentiments	-0.00038	-0.00362
IBES_updown_target	analyst consensus stock price	analyst sentiments	0.03101	0.26932

Table 3: A Summary of Features Used in PEM

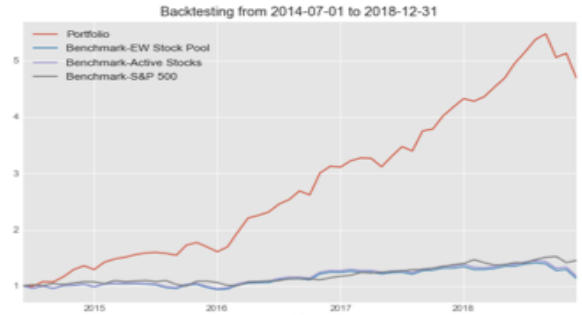
5 Backtesting Results

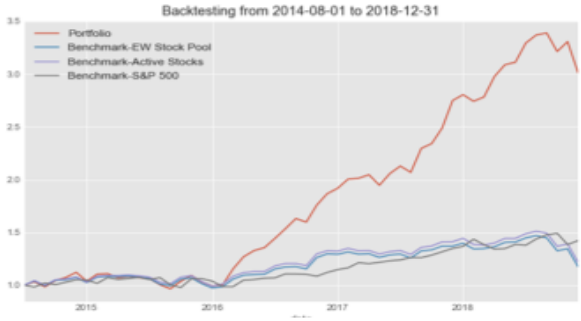
The backtesting results mirror how well the four connected modules work in tandem with each other, as well as how desirable RL-ML-DL-driven active portfolio management versus passive investing.

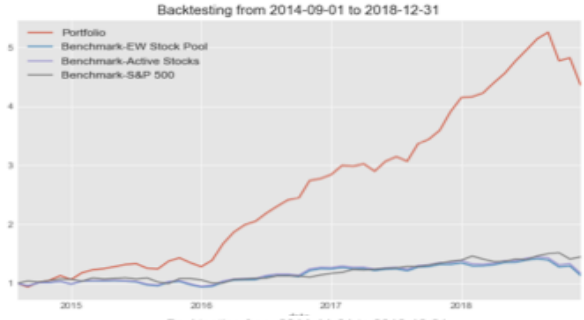
When measuring our portfolio performance, we used three chief benchmarks. The 1st one is evaluated based on the equal-weighted return of our stock universe plus an assumed yearly dividend yield of 3%. The 2nd one only holds the stocks that are always active in the testing window, excluding those involved in delisting, acquisition, or bankruptcy, which nearly cut the half size of the stock pool. The 3rd one is the S&P500 index, which gives a sense of how the overall market performed historically.

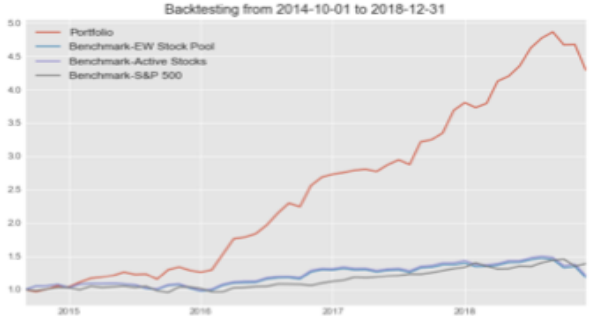
5.1 Results for Deep Neural Networks

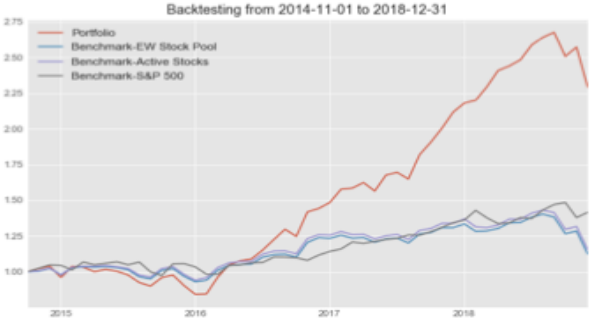
To stay objective and compare with the last group's results, we backtest during the out of sample period with 12 different start months (Jul 2014 to Jun 2015) and the same end month (Dec 2018), so that each test starts off with different initial holdings. We present in the following figures and Table 4 & 5 the performance metrics of each test plus two representative equity curves plots.

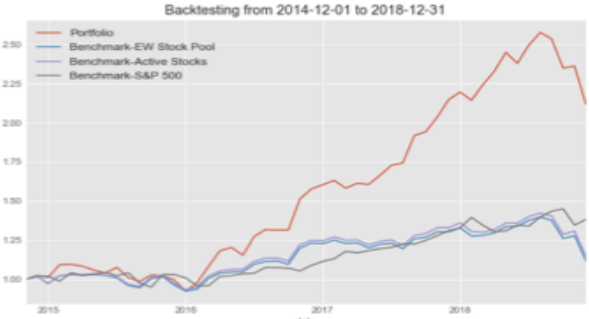

Backtesting from 2014-07-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500

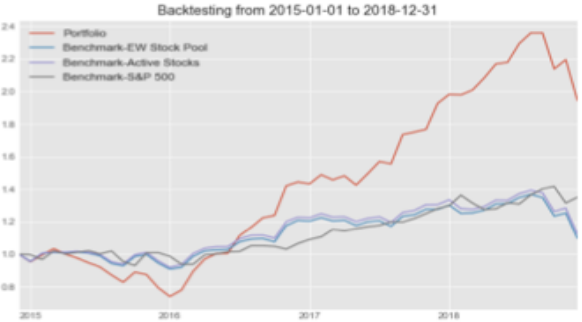

Backtesting from 2014-08-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500

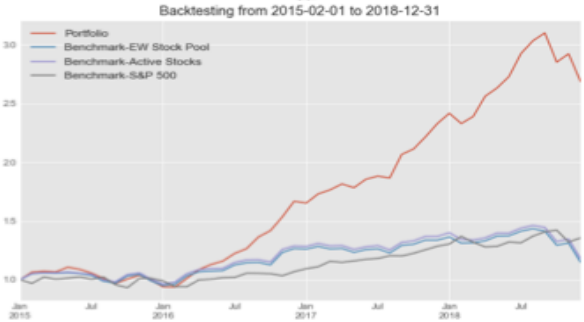

Backtesting from 2014-09-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500

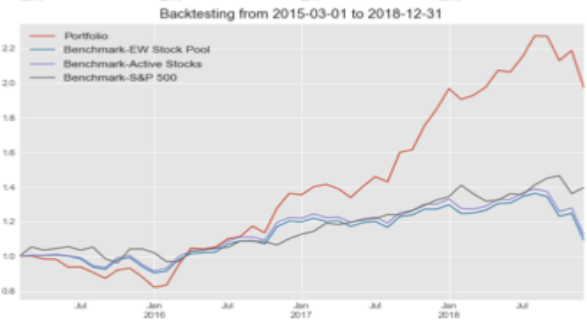

Backtesting from 2014-10-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500

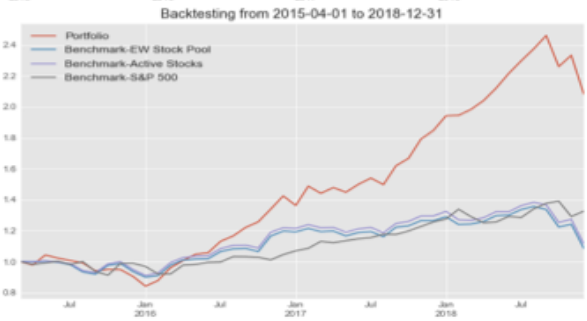

Backtesting from 2014-11-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500

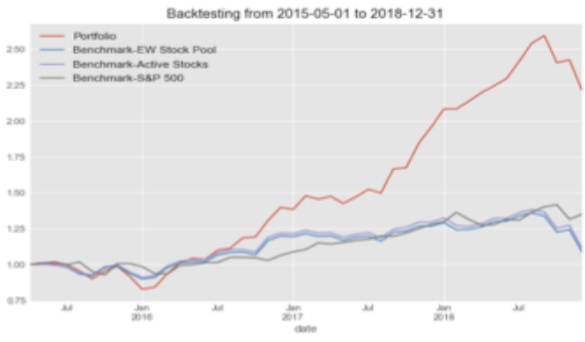

Backtesting from 2014-12-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500

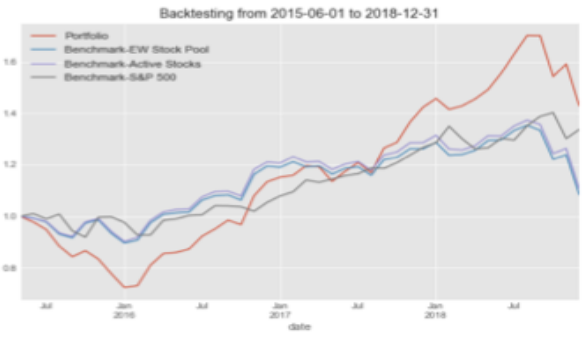

Backtesting from 2015-01-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500


Backtesting from 2015-02-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500


Backtesting from 2015-03-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500


Backtesting from 2015-04-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500


Backtesting from 2015-05-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date


Backtesting from 2015-06-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

Start Month	End Month	Annu. Excess Return	Max REL. Drawdown	Information Ratio
2014-07-01	2018-12-31	0.38216211	0.02514015	4.40535299
2014-08-01	2018-12-31	0.24614637	0.07211712	2.70046921
2014-09-01	2018-12-31	0.37588024	0.02038439	3.86799585
2014-10-01	2018-12-31	0.36822424	0.07985999	3.04344475
2014-11-01	2018-12-31	0.19203507	0.11788937	2.17143924
2014-12-01	2018-12-31	0.17364871	0.06482184	1.98222275
2015-01-01	2018-12-31	0.15762589	0.20360986	1.52326632
2015-02-01	2018-12-31	0.25091224	0.06685421	3.05083377
2015-03-01	2018-12-31	0.17058412	0.09132121	1.89496755
2015-04-01	2018-12-31	0.19399471	0.1195286	2.06256888
2015-05-01	2018-12-31	0.21859608	0.09972289	2.35467253
2015-06-01	2018-12-31	0.08200065	0.19440255	1.05801174

Table 4: Backtesting Results starting from Jul 2014 to Jun 2015

The above result validates that our upgraded portfolio management has the potential to beat the market: in all the twelve tests epoch, our portfolio excess profit surpasses the benchmark whereas the last group backtested in eleven out of the twelve scenarios, which the portfolio return defeats the benchmark yield.

5.2 Results for Decision Tree and Random Forest

Furthermore, we train the decision tree and random forest algorithms on a rolling yearly basis, which improves the training and test accuracy significantly.

To demonstrate the rolling window analysis for predictive performance, we set the number of increments between successive rolling windows to 1 year. (e.g. '2015/01/01' to '2015/12/31') Subsequently, we partition the entire data set into 4 subsamples. The first rolling window contains observations ranging from '2009/03/30' to '2014/12/31', the second rolling window contains observations for period '2010/01/01' through '2015/12/31', and so on.

Again, we backtest during the out of sample epoch with 12 different start months (Mar 2015 to Feb 2016) and the same end month (Dec 2018), so that each test starts off with different initial holdings.

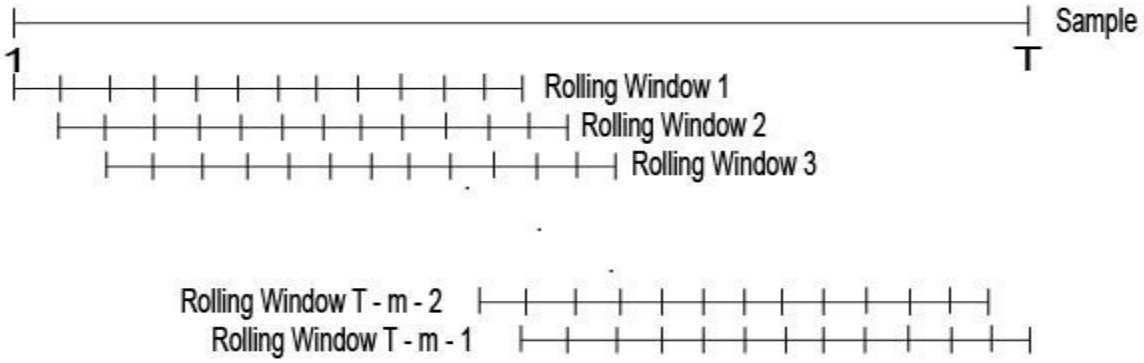
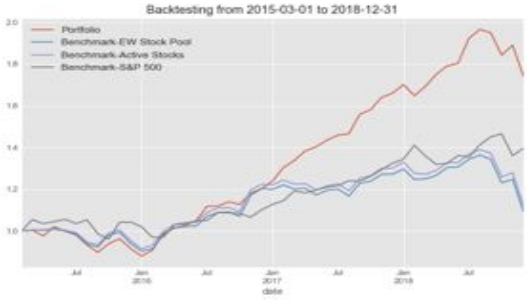
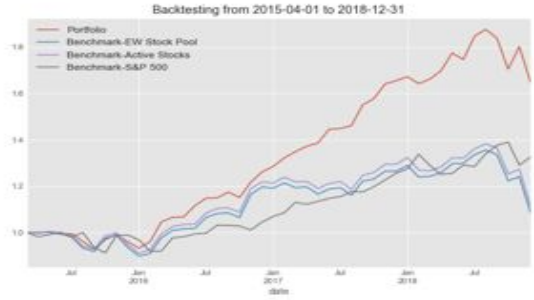


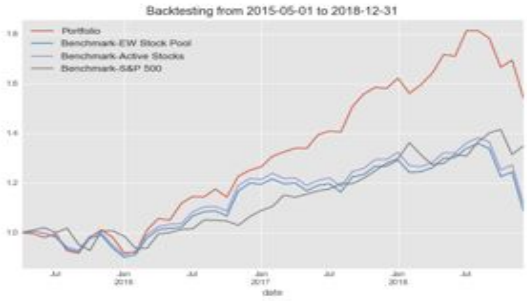
Figure 11: Rolling Window Analysis for Predictive Performance

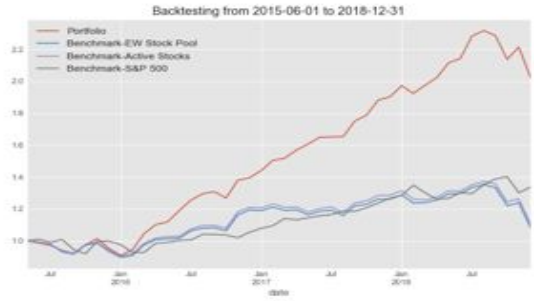
Start Month	End Month	Annu. Excess Return	Max REL. Drawdown	Information Ratio
2015-03-01	2018-12-31	0.13226504	0.0555254	1.94021032
2015-04-01	2018-12-31	0.12051176	0.04590847	1.92272487
2015-05-01	2018-12-31	0.10199538	0.04736637	1.57003737
2015-06-01	2018-12-31	0.19472229	0.03305267	3.33408052
2015-07-01	2018-12-31	0.31923129	0.0563295	3.82527396
2015-08-01	2018-12-31	0.18127337	0.06875201	2.60456042
2015-09-01	2018-12-31	0.04275754	0.30158605	0.37093871
2015-10-01	2018-12-31	0.10525163	0.15036454	1.22827399
2015-11-01	2018-12-31	0.0776776	0.17204656	0.85763031
2015-12-01	2018-12-31	0.34071971	0.09534496	3.310057
2016-01-01	2018-12-31	0.0484374	0.24371892	0.48175171
2016-02-01	2018-12-31	0.13936248	0.12204171	1.76483332

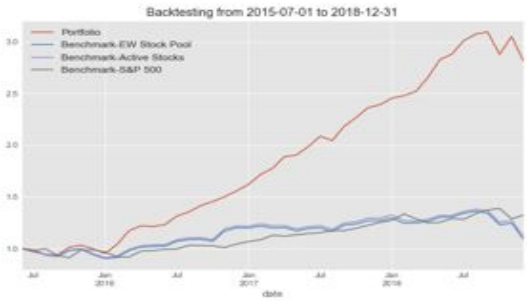
Table 5: Backtesting Results starting from Mar 2015 to Feb 2016

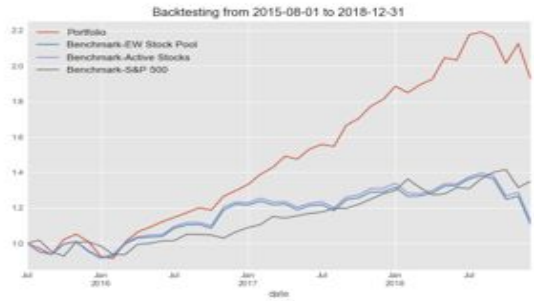

Backtesting from 2015-03-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

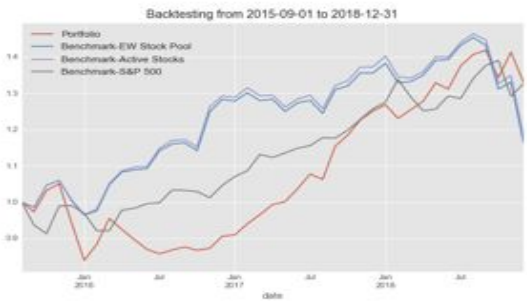

Backtesting from 2015-04-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

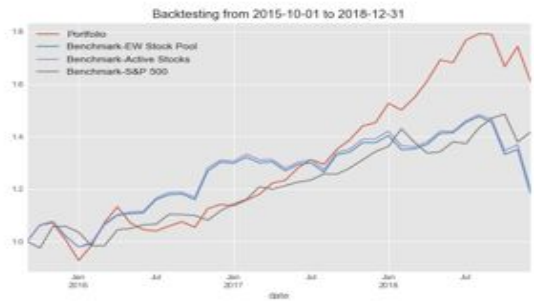

Backtesting from 2015-05-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

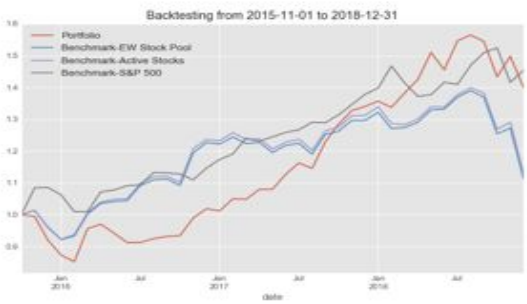

Backtesting from 2015-06-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

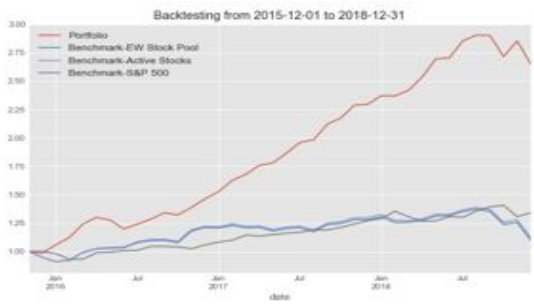

Backtesting from 2015-07-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

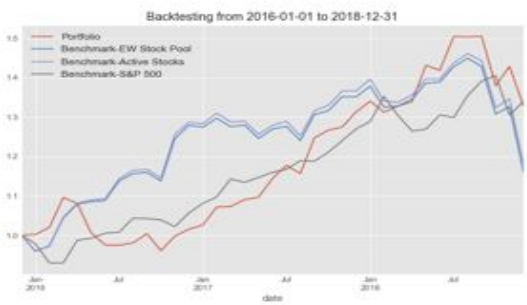

Backtesting from 2015-08-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

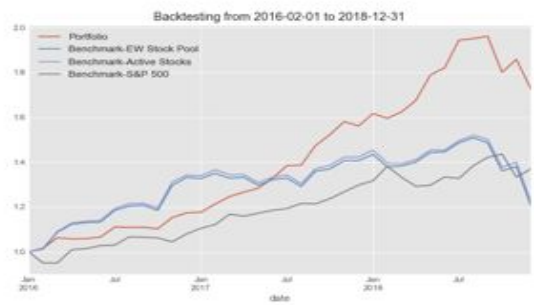

Backtesting from 2015-09-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date


Backtesting from 2015-10-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date


Backtesting from 2015-11-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date


Backtesting from 2015-12-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date


Backtesting from 2016-01-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date


Backtesting from 2016-02-01 to 2018-12-31
Portfolio
Benchmark-EW Stock Pool
Benchmark-Active Stocks
Benchmark-S&P 500
date

6 Conclusion and Prospect

Throughout the paper, we outline four-part architecture individually and how these four modules function together to form a fully automated portfolio optimization system. Besides, we present the corresponding improvement we have made. Powered by machine learning and deep learning algorithms, the individual modules have yielded promising preliminary results. The results of the portfolio backtesting with 21 out of 24 testing epochs prove that the system has the potential to beat the market and generate the alpha.

While the overall results for this system are promising and satisfying, there are certain limitations and some interesting future extensions. Firstly, this portfolio optimization system is used to construct a portfolio of stocks. In the future, we can bring this project to a more general situation. For example, to construct a portfolio of mutual funds or ETFs. Secondly, we could explore various ways to set the initial weight for the portfolio. For the last group, they just set equal weights for each stock when constructing the portfolio. We have improved that by applying the Markowitz Theory. Besides, we could try more advanced allocation methods such as Black–Litterman model, or maximize the sharpe ratio and compare them with each other to find the most effective way to set the initial weights. Thirdly, in our project, all the features we selected to perform the classification and prediction are numerical variables. We could add some NLP work. For example, do text analysis for the Trumps' tweets or stock holders' comments. Through adding these "content" features, we could get a more accurate prediction for the market performance and stock price. Lastly, as markets in turmoil amid the coronavirus outbreak, we could have further discussion on the economic effect of coronavirus.

Appendix: Morgan Stanley Trade Log - Data Analysis

1 Industry analysis

Thanks to the support of the fund manager of Morgan Stanley, we got the trade log data from 2016-02-11 to 2020-04-07. In this part, we will mainly analyze the trade log in terms of the industry, return rate and detail transactions.

The diversified stocks in the market are usually classified into 11 major sectors representing key areas of the economy. The Global Industry Classification Standard also known as GICS is the primary financial industry standard for defining sector classifications. It begins with 11 main categories and could be further divided into 24 industry groups, 68 industries, and 157 sub-industries. It defines a coding system which

could label all the companies which are publicly traded in the market. One of the greatest advantages of this system is that it can help both the investors and the fund managers to have a clear picture of a specific industry which they are interested in. And the 11 main sectors are as follows.

- Energy
- Materials
- Industrials
- Consumer Discretionary
- Consumer Staples
- Health Care
- Financials
- Information Technology
- Telecommunication Services
- Utilities
- Real Estate

In our report, we use Yahoo Finance's sector classification which is a little different from the official GICS. Firstly, we calculated the percentage of transactions for different sectors. The result is shown in Figure 1.

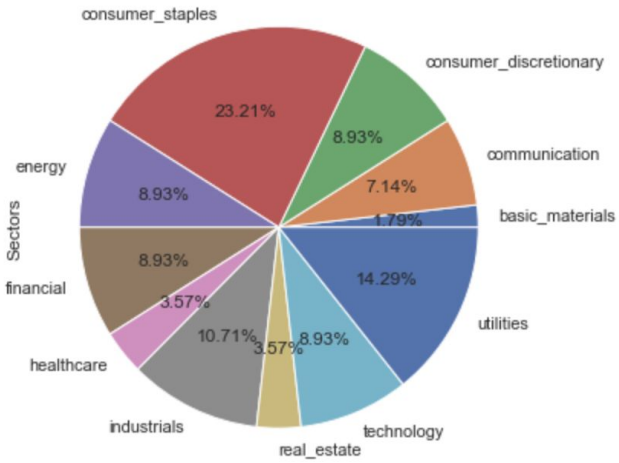


Figure 1

From Figure 1, we can find that from 2016-2020, the portfolio of Morgan Stanley is very diversified, which included all 11 main sectors. Among all the sectors, the Consumer Staple Sector has the largest proportion. And there are few transactions involved in the basic materials, real estate and healthcare sector. However, regarding the healthcare sector, there is a huge difference between various years.(Figure 2 3)

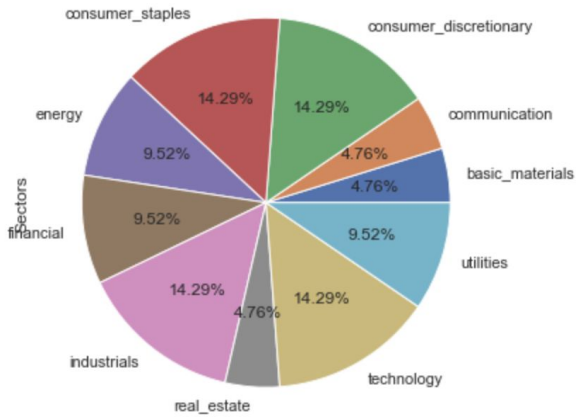


Figure 2 Trades of 2016

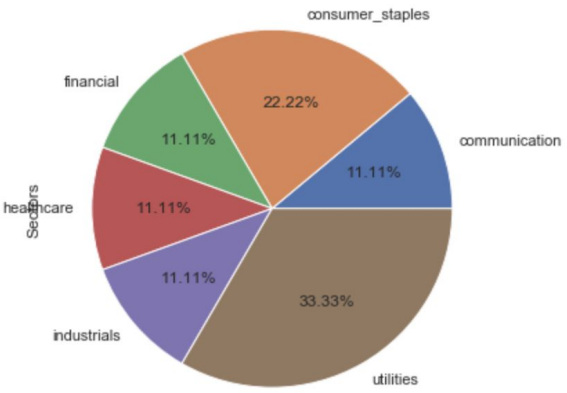


Figure 3 Trades of 2020

The COVID-19 pandemic, also known as the coronavirus pandemic, is an ongoing pandemic of coronavirus disease 2019 (COVID‑19), caused by severe acute respiratory syndrome coronavirus 2 (SARS‑CoV‑2). The outbreak was first identified in Wuhan, China, in December 2019. The World Health Organization declared the outbreak a Public Health Emergency of International Concern on 30 January, and a pandemic on 11 March. As of 16 May 2020, more than 4.56 million cases of COVID-19 have been reported in more than 188 countries and territories, resulting in more than 308,000 deaths. More than 1.64 million people have recovered. With the spread of the virus, more and more people start focusing on the healthcare industry. And this trend also reflected on Morgan Stanley's trade log. In 2016, there were no relevant transactions involved in the healthcare industry. However, when it comes to the beginning of 2020, over 10% of the trades are in the healthcare industry.

2 Trade Time Analysis

As it shows in the trade log, from 2016-2020, KHC(The Kraft Heinz Company) is traded most. Therefore, in this part, we will mainly analyze the trades related to this stock. From Figure 4 we can see the trade history of KHC. From July, 2016 to July, 2017, the fund manager of Morgan Stanley bought the KHC stock at the lowest price on November 10th, 2016 at $81.44. When it comes to the first half of 2018, the price plunged as a result of awful earnings. And this time, the fund manager also had an accurate prediction. He rebalanced the holding of the stocks on February 15th, 2018, when the price firstlt began to plunge. With the steeper competition, changing customer tastes and higher input cost, the stock price of KHC continued to plunge in 2019. On April 17th, 2019, the price hit a record low $32.9. And at this time, the fund manager rebalanced the stocks again.

Figure 4 Historical Stock Price for KHC

And before KHC hit a record low in 2020, the fund manager sold the stock on February 6th, 2020. Generally speaking, the fund manager has an accurate prediction and judgment for the stock price. For the four stages, he firstly bought at a relatively low price, and when the stock price sharply decreased, he rebalanced the stocks twice. And when the company met financial problems, before hitting a record low in 2020, the fund manager sold the stocks wisely. Therefore, we could conclude that selecting a good stock, rebalancing overtime, and catching the right trading time are the three most important things for a strategy. In our report, we also utilized this philosophy to construct our Portfolio Optimization System.

References

[1] Connor Goggins, Kai Wu, Jianqiao Hao, and Yuxuan Mao. Portfolio Optimization System (POS). In Columbia University IEOR E4742 - Deep Learning for Operations Research & Financial Engineering, Dec 2019.

[2] "Rolling-Window Analysis of Time-Series Models." MathWorks, www.mathworks.com/help/econ/rolling-window-estimation-of-state-space-models.html.