Halil Tunçer
CSSM502 Term Project

PORTFOLIO OPTIMIZATION: COMPARATIVE APPROACH

Introduction

In the vast expanse of financial literature, portfolio optimization emerges as a cornerstone discipline, drawing extensive academic scrutiny across various fields. The pursuit of maximizing expected return while minimizing risk, encapsulated in the sophisticated mathematical framework of portfolio optimization, manifests in diverse forms. However, the most important model in this realm remains the Mean-Variance portfolio optimization, a groundbreaking contribution by Harry Markowitz. Markowitz's approach, revered for its adeptness in maximizing returns and mitigating portfolio risk, serves as the bedrock in this domain.

In contemporary finance, the landscape has witnessed a transformative integration of Machine Learning and Deep Learning methods into portfolio management. These advanced techniques play a pivotal role in predicting stock prices, providing investors with the means to meticulously calculate expected returns and associated risks for the variables under consideration. This predictive capability empowers investors in the strategic formation of their portfolios, marking a noteworthy evolution in the field of portfolio optimization.

This project revolves around the primary objective of forecasting individual stock prices and subsequently crafting an optimal portfolio based on these forecasts. This project has two stages: the first aim is to predict the stock prices with classical time series method ARIMA and machine learning Random Forest Regressor model and the second stage involves the constructing optimal portfolios based on predictions. The other aim of this study is to compare the Mean - Variance portfolio or Markovitz portfolio which (benchmark portfolio) with prediction-based portfolios.

Literature Review

The relationship between portfolio optimization and machine learning has been a flourishing field. There has been keen interest in using deep learning and machine learning methods in quantitative finance and portfolio optimization. This literature can be divided into two strands: the first strand of the literature focuses on using machine learning techniques to forecast stock returns and form the portfolios using traditional techniques. The other strand of literature focuses on how machine learning methods can be used to assign weights to stocks rather than predict the stocks. For the first group of papers, Ma et al. (2021) uses deep learning and machine learning methods first to pre-select from a pool of stocks and then try to forecast selected stocks to form a portfolio. They compare their ML portfolio with ARIMA forecasted portfolio and find out that their ML portfolio beats classical time series forecast portfolio in terms of return. Chen et al. (2021) uses IFAXGBoost model to predict stock prices in Shanghai stock exchange and uses predicted price to form mean variance portfolio. They find out that their method is superior to no prediction portfolio in terms of risk and return. For the second strand of literature, Bane et. al. (2016) they introduce machine learning algorithms to solving constrained optimization problem. They use regularization and cross validation method to choose the optimal weight for their portfolios. Awoye (2016) applies graphical Lasso method to portfolio covariance estimation and choses the optimal weight accordingly. He found out that their strategy performs well both in real stock prices and synthetic stock prices. My work closely related to the first strand of literature. I contribute literature using Random Forest regressor as a machine learning method.

Data

To construct portfolio, I choose from S&P 500 stocks. There are various reasons why I choose from the US stock market is that the first reason is that US stock market is deep and very liquid market, there are no market microstructure inefficiencies in the stock prices. The other reason

is that I can find the stock prices easily and up-to-date way. The selected stocks for portfolio formation encompass four stocks: "Coca Cola," "J.P. Morgan," "P&G," and "MSFT." The reason why I choose these stocks is that I believe that they are mor or less representative the whole stock market, they are from the different sectors and leader in their sectors.

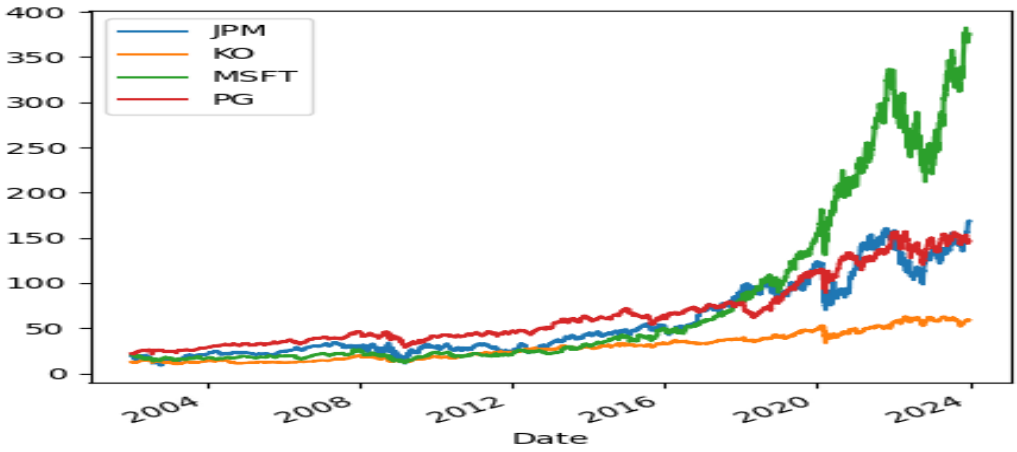


Figure 1: Sample for Stocks

The dataset utilized for analysis spans daily stock prices, commencing from January 1, 2002, and extending through December 31, 2023. I acquire stock prices from yahoo finance package (yfinance). The overarching goal is to leverage these historical stock price trends to inform robust predictions, ultimately guiding the strategic construction of an optimal and well-balanced portfolio.

Model

Benchmark Model:

In pursuit of constructing an optimal portfolio from the four selected stocks ("Coca Cola," "J.P. Morgan," "P&G," "MSFT"), I employ Harry Markowitz's renowned portfolio optimization technique as a foundational methodology. The crux of this approach involves strategically selecting asset weights to maximize returns while minimizing risk. To fine-tune the weight allocation, I used Monte Carlo Simulation. This simulation methodology becomes instrumental

in determining the optimal weights by prioritizing the highest Sharpe Ratio, a metric that juxtaposes expected return against expected risk.

Initially, the quest for optimal weights involves solving a constrained optimization problem for a benchmark model, providing a rigorous foundation for portfolio construction. In our constrained optimization problem, we try to maximize the Sharpe ratio with respect to weights

```
log_mean = log_ret.mean() * 252
cov = log_ret.cov() * 252
def get_ret_vol_sr(weights):
    weights = np.array(weights)
    ret = log_mean.dot(weights)
    vol = np.sqrt(weights.T.dot(cov.dot(weights)))
    sr = ret / vol
    return np.array([ret, vol, sr])
# Negate Sharpe ratio as we need to max it but Scipy minimize the given function
def neg_sr(weights):
    return get_ret_vol_sr(weights)[-1] * -1
# check sum of weights
def check_sum(weights):
    return np.sum(weights) - 1
# Constraints for the optimization problem
cons = {'type':'eq','fun':check_sum}
# bounds on weights
bounds = ((0,1),(0,1),(0,1),(0,1))
# initial guess for optimization to start with
init_guess = [.25 for _ in range(4)]
opt_results = optimize.minimize(neg_sr, init_guess, constraints=cons, bounds=bounds, method='SLSQP')
```

that sum up to one. For the constrained optimization problem, I used the Scipy optimize function to solve the optimization problem.

When we solve this optimization problem, we get weights for each stock: PG has 0.1%, KO has 10.2% weight, MSFT has 35.87%, JPM has 53.82% weight.

Remarkably, the results obtained through this model align closely with those derived from the Monte Carlo Simulation. The Monte Carlo Simulation is a simulation technique that randomly assign weights to each stock. I run 10.000 Monte Carlo Simulation and choose the portfolio that has the maximum Sharpe Ratio, which is the expected return divided by expected risk

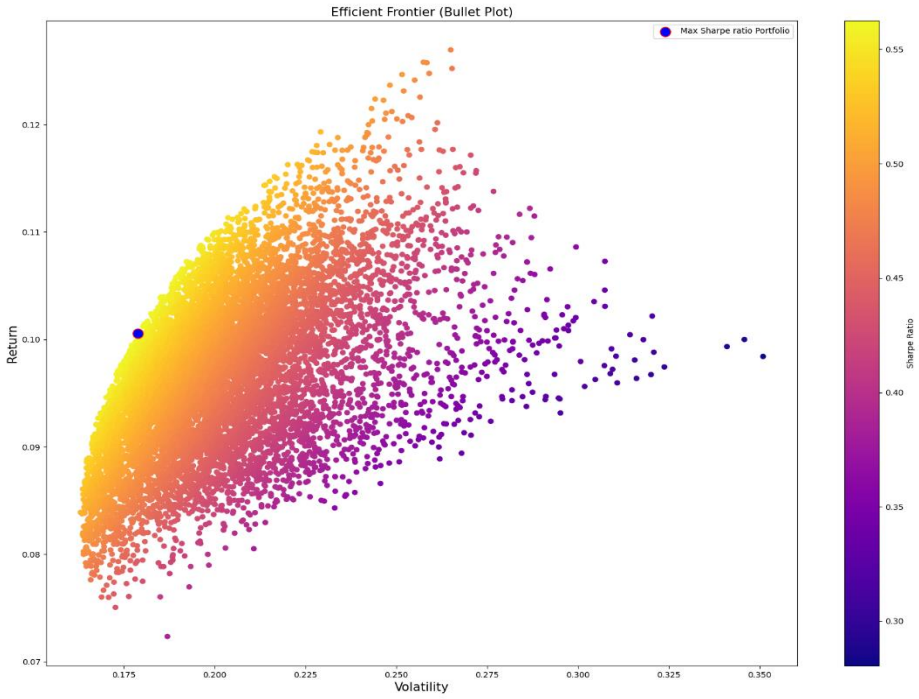


Figure 2: Mean-Variance Portfolio Monte Carlo Simulation

The figure 2 shows 10.000 portfolio as a result of Monte Carlo Simulation for the actual prices. The blue dot stands for the portfolio that has the maximum Sharpe Ratio, the weight of each stock in this portfolio as follows: P&G has 1.6%, KO has 12.83%, MSFT has 35.98%, JPM has 51.03% weights. So, the results are very similar, even we can say that they are the same. Encouraged by this consistency, I persist in leveraging the simulation technique to ascertain the most advantageous weight distribution for the selected stocks. This dual-pronged approach ensures a robust and well-informed selection process, highlighting the synergy between

traditional optimization methodologies and innovative simulation techniques in portfolio management.

AutoRegressive Integrated Moving Average Model:

I also integrate the ARIMA (Autoregressive Integrated Moving Average) process for forecasting stock prices, a classic time series econometrics tool. Following the standard practice in forecasting, I meticulously split the dataset into training and test sets, opting for an 85:15 proportion to ensure a balanced evaluation. Importantly, the test period initiates from September 10, 2020, strategically excluding the Covid-19 period to prevent potential forecasting challenges.

Figure 3: Train Test Split

The figure 3 shows the train test split for J.P. Morgan stocks. On the y-axis, you see the log price of the JPM. The train-test split is the same for all the stocks and it is also the same in Machine Learning model.

Given the inherent non-stationarity of stock prices characterized by their random walk nature, a vital pre-processing step involves transforming the data. We can also check whether the data is stationary or not by looking at Augmented Dickey Fuller statistics. For the JPM stationary test, we have the results as follows:

Results of dickey fuller test	
Test Statistics	0.614378
p-value	0.987965
No. of lags used	33.000000
Number of observations used	5503.000000
critical value (1%)	-3.431539
critical value (5%)	-2.862065
critical value (10%)	-2.567050
dtype: float64	

Table 1: Test for Stationary

Our test statistics is below the critical values; hence we cannot reject the null hypothesis that JPM is non-stationary. I also did the ADF test for each stock and find that other stocks are non-stationary as well. Recognizing this, I take the logarithm of the stock prices, resulting in log prices, a technique that aids in achieving stationarity.

For each individual stock, I leverage the AutoArima model in Python to systematically determine the optimal autoregressive (p), moving average (q) parts, and the necessary differencing (d). In classical time series models, taking logarithm of the data and taking the first difference of the data is the way to make data stationary. This package tells us the optimal lag length and whether we should take difference to make the data stationary.

```
Best model:  ARIMA(0,1,5)(0,0,0)[0]
Total fit time: 64.426 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 4703
Model:               SARIMAX(0, 1, 5)   Log Likelihood               10838.492
Date:                Tue, 09 Jan 2024   AIC                          -21664.985
Time:                        13:46:49   BIC                          -21626.250
Sample:                             0   HQIC                         -21651.366
                                - 4703
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.1100      0.006    -19.758      0.000      -0.121      -0.099
ma.L2          0.0004      0.006      0.060      0.952      -0.012       0.013
ma.L3         -0.0268      0.007     -4.071      0.000      -0.040      -0.014
ma.L4         -0.0444      0.007     -6.781      0.000      -0.057      -0.032
ma.L5         -0.0205      0.007     -3.119      0.002      -0.033      -0.008
sigma2         0.0006   4.13e-06    141.045      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):             46174.36
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               0.75   Skew:                            -0.02
Prob(H) (two-sided):                  0.00   Kurtosis:                        18.35
===================================================================================
```

Table 2: Model Search

For JPM example, the autoarima package searach over different ARIMA models and find the best model according to Akaike Information Criteria, the lowest the score the better the model. For JPM, the best model is (0,1,5). This model says that we should take difference to make the data stationary and we can model the log price of JPM stock by using constant plus independent random noise and five lag of the random noise process. After finding the optimum model, we set up the model accordingly.

SARIMAX Results

Dep. Variable:	JPM	No. Observations:	4703
Model:	ARIMA(0, 1, 5)	Log Likelihood	10839.131
Date:	Tue, 09 Jan 2024	AIC	-21664.261
Time:	13:47:01	BIC	-21619.071
Sample:	0	HQIC	-21648.373
	- 4703		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
x1	0.0003	0.000	1.069	0.285	-0.000	0.001
ma.L1	-0.1103	0.006	-19.581	0.000	-0.121	-0.099
ma.L2	0.0001	0.006	0.019	0.985	-0.013	0.013
ma.L3	-0.0270	0.007	-4.104	0.000	-0.040	-0.014
ma.L4	-0.0447	0.007	-6.809	0.000	-0.058	-0.032
ma.L5	-0.0208	0.007	-3.144	0.002	-0.034	-0.008
sigma2	0.0006	4.13e-06	141.071	0.000	0.001	0.001

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	46159.84
Prob(Q):	0.96	Prob(JB):	0.00
Heteroskedasticity (H):	0.75	Skew:	-0.02
Prob(H) (two-sided):	0.00	Kurtosis:	18.35

Table 3: Best Model

After setting up the model, we forecast in-sample prediction and out of sample forecasting for test period of JPM.

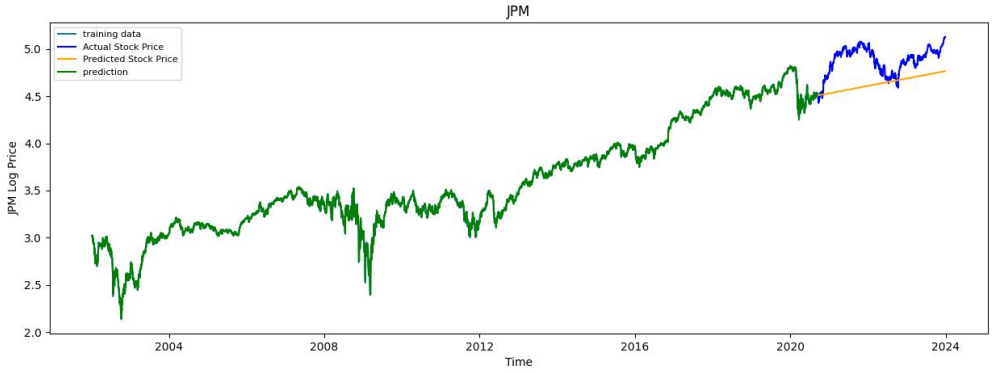


Figure 4: Arima Forecast Results for JPM

I followed the same process for each stock. Once armed with the optimal ARIMA(p,d,q) parameters, I construct forecast equations for each stock. For Coca Cola, the optimum model is (1,1,0). The model says that movement of KO can be explained by its own first lag only. For MSFT, the best model is the (0,1,1), we can only explain the movement of MSFT by first lag of independent random noise model. You can see the forecast graphs in the appendix.

Following the individual stock forecasts, I calculate the expected return and variance of the portfolio. To refine the portfolio's weight distribution, I employ the Monte Carlo Simulation methodology, prioritizing the highest Sharpe ratio as the guiding metric.

Figure 5: Arima Monte Carlo Simulation Results

This iterative process ensures that the portfolio is not only optimized based on historical trends but also considers the intricacies of forecasting and the dynamic nature of financial markets.

Random Forest

In the realm of Machine Learning, I harnessed the power of the Random Forest Regressor to predict stock prices. This methodology operates by training an extensive ensemble of decision trees on a dataset with inherent tree-like structures, subsequently averaging their predictions. The rationale behind choosing the Random Forest method lies in its adaptability to the daily stock price dataset spanning from 2002 to 2023. Leveraging information from daily open, close,

high, and low prices, the Random Forest Regressor effectively partitions stock prices into diverse branches, culminating in an averaged prediction. This strategy is particularly robust for forecasting stock prices, considering the multifaceted nature of the data. The other advantage of Random Forest Regressor is its avoidance to overfit the model.

In this project, the focus of prediction was on the adjusted close price of each stock. The independent variables used for prediction included low, high, and closed prices on a given day. To maintain consistency with earlier models, each variable underwent a logarithmic transformation. I also pre-process the data with MinMaxScaler function to restrict the values between 0 and 1. For each stock, the Random Forest Regressor was meticulously tuned by Randomized Search Cross Validation, and hyperparameters were optimized using the same train-test split as employed in the ARIMA model. For hyperparameters tuning, I used the following parameters: estimator to control number of trees, max_features to control number of features in every split, max_dept, min_samples_split, min_samples_leaf, boostrap to sample data points. I make these data processing in every stock and forecast the stock prices. For example, you can find the JPM Random Forest Forecast:

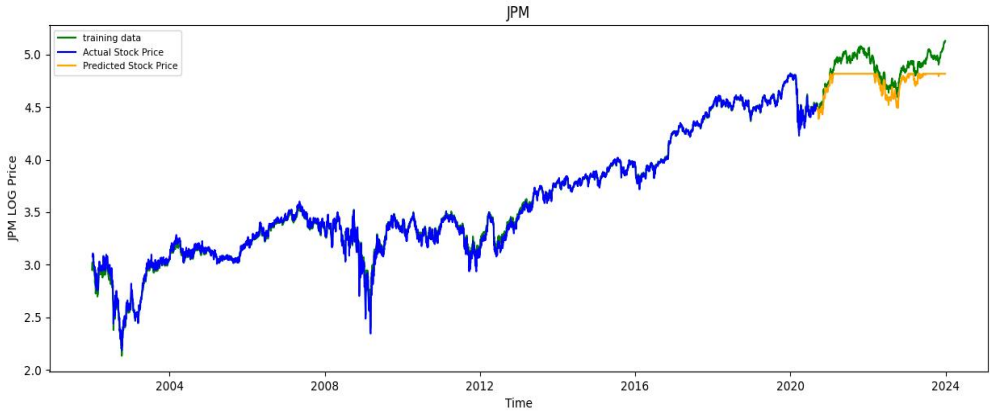


Figure 6: Random Forest Forecast for JPM

Post-prediction, I computed the annualized expected return and variance (risk) for each stock. Employing the Monte Carlo Simulation, I determined the optimal portfolio configuration based on the highest Sharpe Ratio.

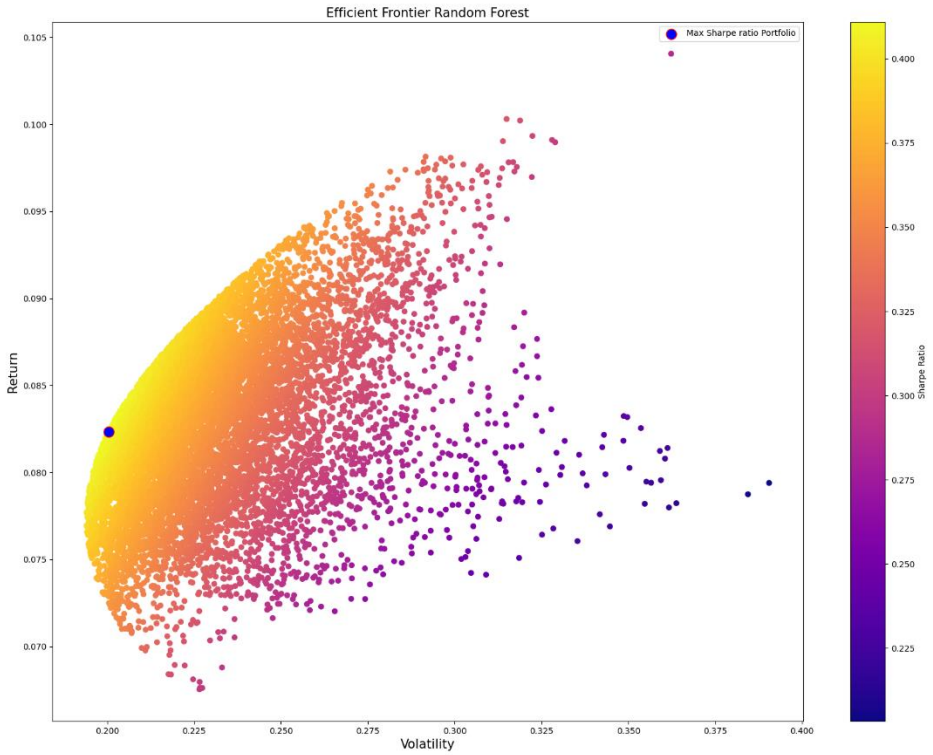


Figure 7: Random Forest Monte Carlo Simulation

Results

When we compare the forecast results of two model, we can look at the mean squared error and root mean squared error of the models for each four stocks, mean squared error stand for the average of the squared difference between original and predicted variables. RMSE is just the taking the root of MSE. So, MSE is the variance of the residuals, RMSE is the standard deviation of residuals.

	JPM	KO	PG	MSFT
MSE ARIMA	0.0777	0.0102	0.0085	0.0270
MSE FOREST	0.1264	0.0910	0.0861	0.1999
RMSE ARIMA	0.2788	0.1010	0.0922	0.1645
RMSE FOREST	0.1396	0.0987	0.1050	0.2479

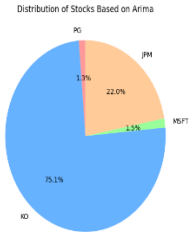
When we look at RMSE and MSE stock by stock for two models, we can say that ARIMA better forecasts all stocks better than Random Forest. In terms of RMSE, except P&G, RMSE of the random forest model is low.

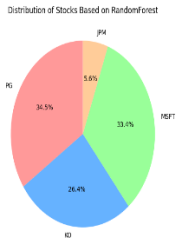
When we look at the optimal portfolio's return, variance, and Sharpe ratios after Monte Carlo Simulation (n=10000), we can have a better sense of which method has the superior performance in terms of portfolio performance.

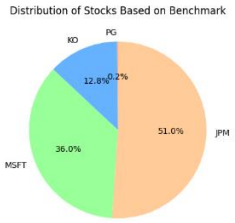
	Expected Return	Expected Variance	Sharpe Ratio
Markovitz Benchmark	0.1005	0.1787	0.5625
ARIMA Model	0.1100	0.1726	0.6373
Random Forest Model	0.0820	0.2008	0.4084

Our benchmark model portfolio performance's annualized expected return is around 10% and expected variance is around 0.1787, so the Sharpe ratio is around 56%. Based on Arima forecast, annualized return is around 11% and expected variance is around 0.1726 and Sharpe Ratio is 0.63. Based on Random Forest model, the expected annual return is 8%, the expected variance is 0.20 and the Sharpe ratio is 0.41. It seems that Arıma forecast portfolio has the highest Sharpe ratio, lowest variance, and the highest expected return. It outperforms the other portfolios.

We can also look at the how these models choose the optimum weights of four stocks, we can say that in benchmark model, more than 50% of the portfolio is allocated to JPM, 36% of the portfolio is allocated to MSFT and the rest is distributed KO and P&G, 12.8% and 0.2% respectively.







The highest share given in ARIMA forecast portfolio is KO with about 75.1%, then JPM received 22% weight and MSFT and received around 1.5% and 1.3% respectively.

When we look at the portfolio based on Random Forest, P&G and MSFT has almost equal weight in the portfolio, their weights around 34%, KO received the 26.4% and JPM received 5.6% share.

So far, we have compared the optimum portfolios in terms of their annualized return, when we look at the cumulative log returns of the portfolio starting from 2002 to 2024, we can see that up until the test period, benchmark model and returns of Random Forest model goes almost hand in hand, sometimes benchmark model beats forest model or vice versa. ARIMA portfolio

is being outperformed by two other models up until 2018 and after 2018 it beats the benchmark portfolio and ARIMA portfolios. During the test period, ARIMA portfolio is the best performer.

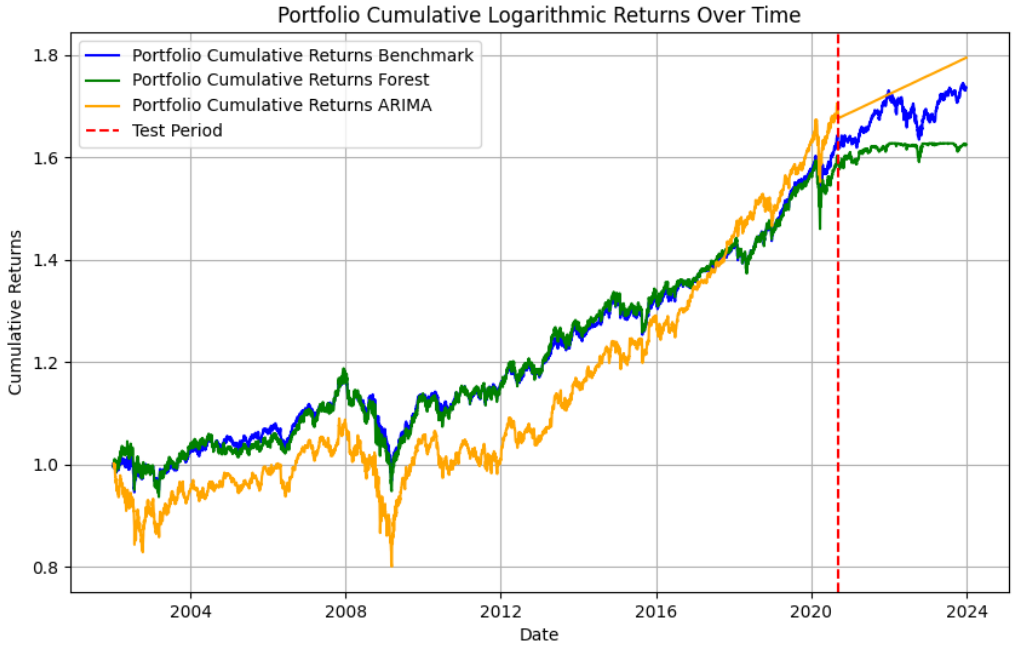


Figure 8: Portfolio Cumulative Logarithmic Returns

I think that it is because of the ARIMA forecast a linear trend for stocks, it affects the return of the portfolio. Benchmark model performs well, even it outperforms ARIMA model for a short time. Random Forest model is the worst performer, it has the lowest variance and has the least drawdown in case of a sell-off but in terms of return, it performed worst.

The cumulative logarithmic return of the ARIMA portfolio is around 80%, if you invest in 1$ in 2002 in ARIMA portfolio, you get 1.8$. In normal return, it means 547% increase in your portfolio. If you invested 1$ in 2002, you get 5.47$ at the end of the period. The cumulative log return of the benchmark model is around 70% , in normal terms it corresponds to 500%. If you invested in benchmark model 1$, you get 5$ roughly at the end of the period. The cumulative log return of the random forest model is around 64% and it has the lowest return amongst three portfolios.

Conclusion

In this study, I try to use Machine Learning method for portfolio optimization problem and test how it performs against the traditional methods. This study has three models: the first one is the Markovitz mean variance model that is used as benchmark model, the second model is ARIMA model in which I first forecasted the stock prices and then form the portfolios, third model is the Random Forest Regressor Model with which I forecasted stock prices and then formed the optimal portfolio through Monte Carlo Simulation. When we look at the results, we can say that ARIMA model is better at forecasting test data compared to Random Forest. When we compare the portfolio returns, Random Forest portfolio returns are very similar to benchmark portfolio returns and outperforms the ARIMA portfolio returns in the training part of the sample. However, in the test part of the sample, Random Forest portfolio is beaten by ARIMA and Benchmark model. In cumulative terms, Random Forest portfolio brings around 64%, while ARIMA and Benchmark portfolios bring logarithmic 80% and 70% respectively.

The reason why Random Forest Regressor fall behind other models may be related to its forecast performance. Random Forest Regressor may be improved by adding other predictors to predict the stock prices well, with just only price predictors it may not create the necessary splitting in trees. Another factor contributing to the challenges in forecasting accuracy is the time series data, which poses difficulties not encountered by the ARIMA model. ARIMA, being a specialized time series model, is explicitly designed to handle the intricacies associated with time-dependent data, giving it a comparative advantage in forecasting accuracy over models that may struggle with the time series dimension.

This research may be improved by adding machine learning methods that are more oriented to time series forecasting. For example, we can use LTSM model. We can also use macroeconomics variables as predictors to predict stock prices. We can also use machine

learning methods in choosing optimal weights. Instead of Monte Carlo Simulation, we can use clustering methods such as: fuzzy clustering, gradient, K means clustering etc.

BIBLIOGRAPHY

Ban, G. Y., El Karoui, N., & Lim, A. E. (2018). Machine learning and portfolio optimization. *Management Science*, *64*(3), 1136-1154

Ma, Y., Han, R., & Wang, W. (2021). Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, *165*, 113973.

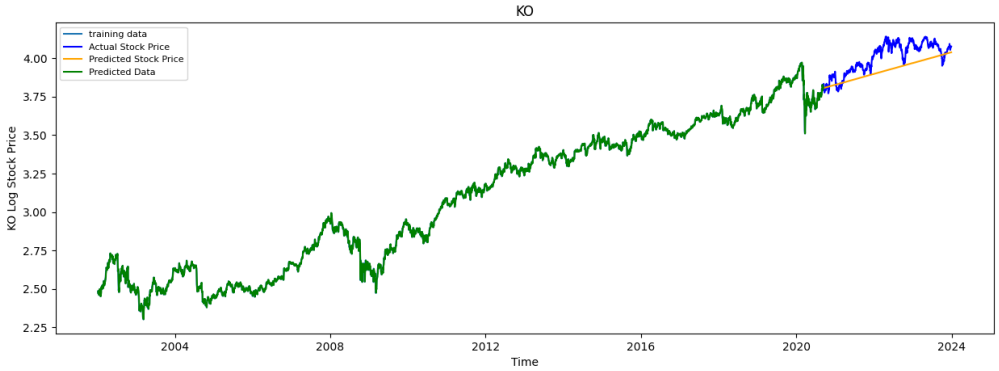
Chen, W., Zhang, H., Mehlawat, M. K., & Jia, L. (2021). Mean–variance portfolio optimization using machine learning-based stock price prediction. *Applied Soft Computing*, *100*, 106943.

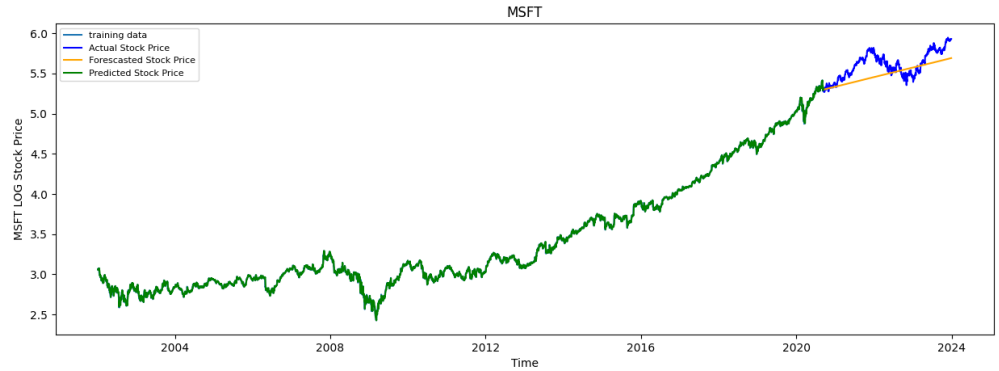
Awoye, O. A. (2016). *Markowitz minimum variance portfolio optimization using new machine learning methods* (Doctoral dissertation, (UCL) University College London).

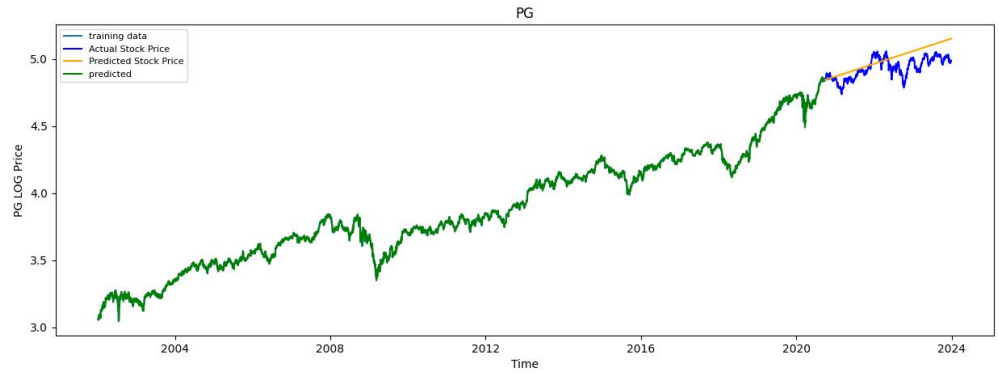
Halil Tunçer
CSSM502 Term Project

APPENDIX

Other stocks' ARIMA forecast results:







Halil Tunçer
CSSM502 Term Project

Other Stocks' Random Forest Forecast Results: