QF2103 Computing for Quantitative Finance: Group 13 Project Report

Members: Choi Yat Long (A0266838L), Chng Phoebe (A0244252M), He Caiying (A0236551H), Jeong Hojae(A0289871H)

Introduction

With Python and Machine Learning (ML) algorithms, we tested various models on stock price predictions and selected two models based on their testing performance. The two models chosen are the AutoRegressive Integrated Moving Average (ARIMA) model and the Long Short-Term Memory (LSTM) model. The trading method we employed for both models is to adopt a long position when the model predicts an increase in stock price and to go short when a drop is predicted, to capture the maximum profit from the price fluctuations.

The performances of these two trading strategies are evaluated using a range of performance metrics. We define a benchmark strategy to be going long for the entire period. By comparing the returns generated from our models and the benchmark return, we gain insights about the effectiveness of our strategies.

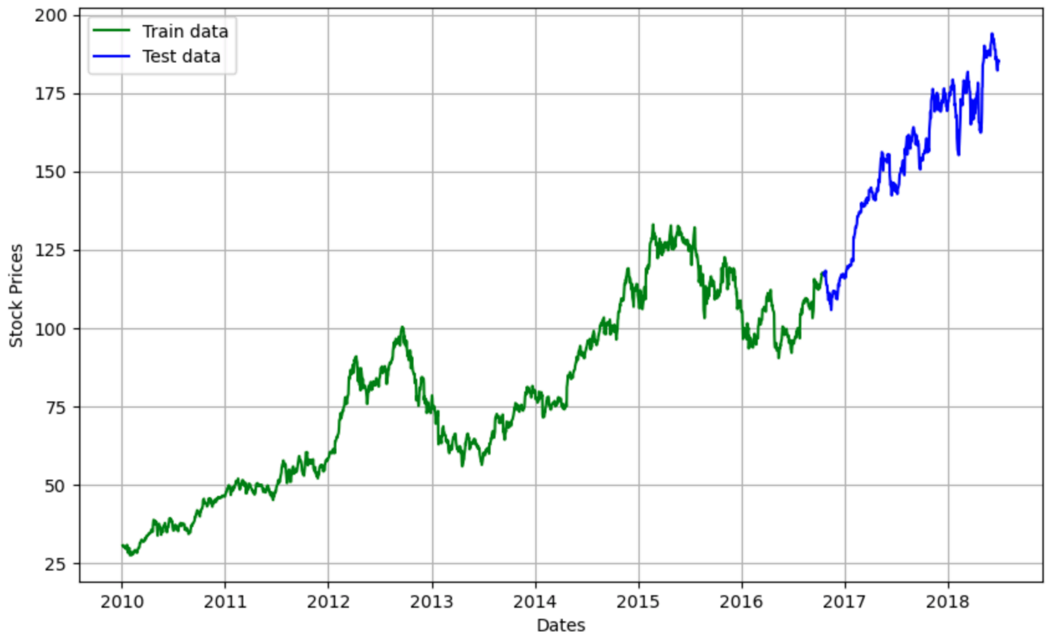
In this report, we will explain our process of exploring the strategies and give a detailed explanation of the logic, implementation, evaluation and challenges of the two selected trading strategies. Lastly, we will discuss further improvements and reflection on the project.

Team Roles and Responsibilities

- Choi Yat Long: strategy selection, exploration and implementation of LSTM strategy
- Chng Phoebe: research and implementation of ARIMA strategy, evaluation of strategy performance
- Jeong Hojae: research on the constraints and improvements of ARIMA, modification of ARIMA strategy with Logistic Regression,
- He Caiying: Code edits support, calculation of different metrics, analysis of future improvements

Detailed Explanation of the Trading Strategy Logic

For each dataset, we split the data into 80% for training and 20% for testing.



<u>Strategy 1: LSTM</u>

Long Short-Term Memory (LSTM) is a variant of Recurrent Neural Network (RNN) which can be used for analysing and forecasting time series data due to its ability to memorise information over relatively long time periods. In contrast to neural networks as described in the lecture, in LSTM and RNN, the output may be feedback as an input to the network with the next input which formulate sequential memory. In addition, it utilises a gate mechanism which consists of an input gate, a forget gate and an output gate. These gates process the information stored in the cell state about the relevant time series, enabling effective prediction for both short-term and long-term time series data.

An LSTM cell integrates three sources of information: the current input sequence, the short-term memory derived from the output of the preceding cell, and the long-term memory inherited from the previous cell state's output. The forget gate evaluates both the current input and the previous short-term memory, discerning which information should be discarded in the long-term memory. Meanwhile, the input gate determines which new information should be incorporated into the cell and used to update the memory. Lastly, the output gate selects the relevant information to be outputted from the current cell state.

<u>Strategy 2: ARIMA</u>

Firstly, we checked if the series is stationary because ARIMA time series analysis generally requires stationary data. The ADF (Augmented Dickey-Fuller) Test can help to determine if the series has a unit root and thus is stationary. The null hypothesis is that the series has a unit root, and the alternate hypothesis is that the series has no unit root. From the results of the ADF test, the p-value is greater than 0.05, hence we cannot reject the null hypothesis. The data is thus non-stationary. Additionally, the increasing mean and standard deviation show that the data is non-stationary.

Stationarity is an assumption of the ARIMA model. Non-stationary data requires differencing to degree d, through the optimal parameters.

The autoregressive integrated moving average (ARIMA) model analyses historical data which identifies trends and patterns, to predict stock prices. It combines three key concepts. First, AutoRegression (AR), this part of the model tries to explain the momentum or the continuity of the series using its previous values. The ‘p’ in ARIMA represents the number of lag observations included in the model, also known as the lag order. Second, Integrated (I) represents the differencing of raw observations to make the time series stationary, meaning the statistical properties of the series like mean and variance are constant over time. The ‘d’ in ARIMA is the number of times the data have had past values subtracted, also known as the degree of differencing. Last, the Moving Average (MA) models the error of the model as a combination of previous error terms. The ‘q’ in ARIMA is the size of the moving average window, also known as the order of moving average.

Next, we used the auto_arima function to calculate the AIC value for different combinations of parameters (p,d,q) and subsequently determine the optimal parameters. The AIC (Akaike Information Criterion) is a measure used to compare different possible ARIMA models. It’s a tool for model selection that balances the complexity of the model against how well the model fits the data. A lower AIC score suggests a better model. Our coding was done to minimise the AIC, and the results for the four stocks (AAPL.O, AMZN.O, INTC.O, MSFT.O) were p=0, d=1, q=0, which means that the best fitting ARIMA model for our stock prices data does not need any AR terms or MA terms and requires differencing once to make the data stationary. In practice, this model would predict future stock prices by essentially looking at the changes in prices from one period to the next, without considering any specific momentum or patterns in the error terms. It’s a simplistic model that assumes the best predictor of tomorrow’s price is today’s price, with adjustments made for the general trend in price changes over time. Therefore, a meaningful trading strategy cannot be devised. To provide the necessary predictive power while maintaining the simplicity of the model, p and q were set to either 0 or 1, respectively. Therefore, the optimal parameters were selected among the choices of (1, 1, 0), (0, 1, 1) and (1, 1, 1) only.

Stock	(0,1,1)	(1,1,0)	(1,1,1)	Optimal Parameters
AAPL	5864.563	5864.599	5866.263	(0,1,1)
AMZN	11492.853	11492.853	11494.879	(0,1,1)
GS	7942.633	7942.513	7940.167	(1,1,1)
INTC	1686.846	1686.842	1688.843	(1,1,0)
MSFT	2862.604	2862.604	2864.607	(0,1,1)

The table above shows that the parameters with the lowest AIC were chosen. In AMZN, (0,1,1) and (1,1,0) had the lowest AIC values. We chose (0,1,1) as we will be using the lag price data that goes into the feature of Logistic Regression (done later to improve the ARIMA model), and the first parameter is related to lagged observations. Similarly, we chose (0,1,1) in MSFT.

Next, we defined the ARIMA model. Using the ARIMA function to fit into a model using optimal p,d,q, and training the model using a rolling fit approach. Initially, the history list is populated with the training data (train). For each step in the test data, a prediction (yhat) is made using the arima_forecast function, which fits an ARIMA model on the current history. After making a prediction, the actual observed value (obs) for that step is added to history to update the model state. This process continues iteratively until predictions are made for the entire test set.

Based on the stock price predicted through the ARIMA model, we took a long position if the predicted stock price was higher than the previous stock price, and vice versa, a short position.

Identification of Key Parameters

Strategy 1: LSTM

- Lagged observations
    - We use the stock prices observed on the previous few days to predict stock prices for the next day
- Learning rate
- Epochs
- Batch
- Hidden Layers
- Stacked Layer

As a variant of neural networks, LSTM comprises different layers of neurons. Hidden layers capture long-term dependency while stack layers capture sequential hierarchical and complex patterns. During training, the dataset will be split into smaller batches and each batch will be sequentially fed into the network. In our model, we didn't adjust the batch size in parameter fine-tuning. Once all batches have been processed, one epoch is complete. After each epoch, the parameters in the neuron, e.g. weight, are adjusted using some algorithm. As the number of epochs increases, accuracy increases but may also lead to overfitting. Increasing the learning rate can lead to faster convergence so that less epoch is needed but it may also lead to overshooting that the model fails to converge to an optimal solution. A low learning rate may, on the other hand, lead to slow convergence. We maintain the default learning rate throughout the project.

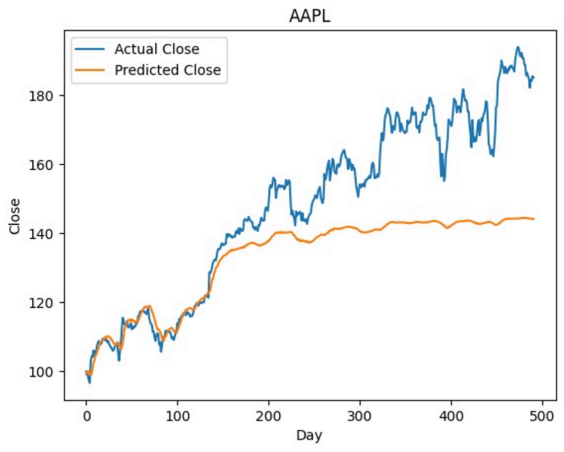
Strategy 2: ARIMA

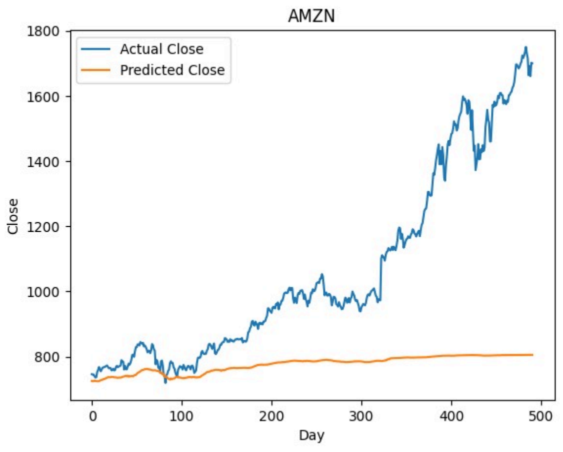
- Lagged observations
  - p: number of lagged observations
- Differences between raw observations
  - d: degree of differencing
- Moving average window
  - q: size of moving average window

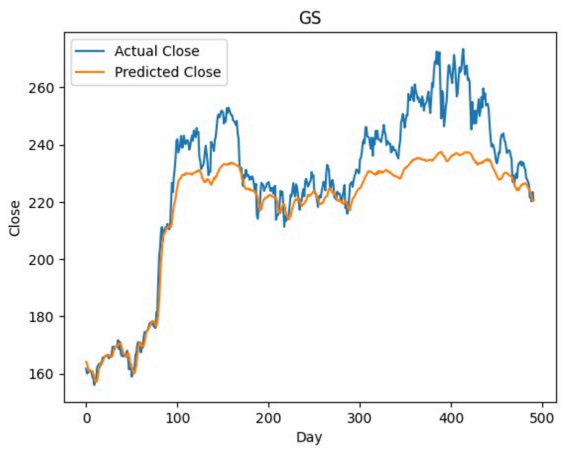
Potential Challenges and Risks of the Trading Strategy

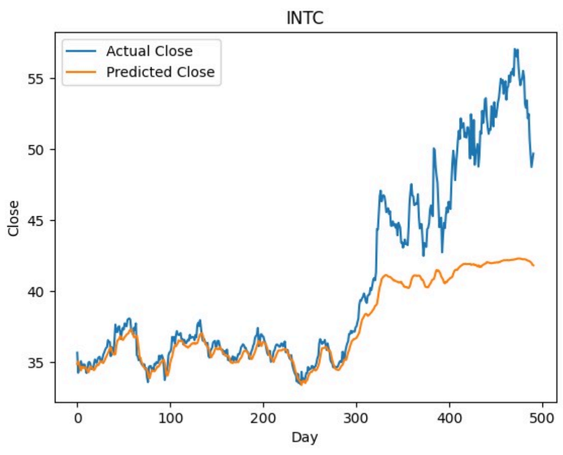
Strategy 1: LSTM

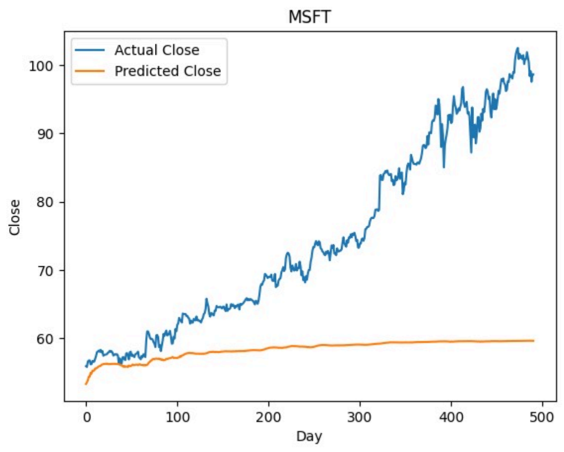
The graph depicted below illustrates the forecasted and actual prices of each stock. Initially, during the testing phase, the predictions closely aligned with the actual prices, but as time progressed, disparities emerged. Notably, the predictions failed to accurately capture fluctuations in stock prices during the training period. Consequently, unlike ARIMA, our trading strategy relied predominantly on forecasting directional changes in prices. Consequently, the accuracy of the trading strategy diminished, resulting in suboptimal yields.



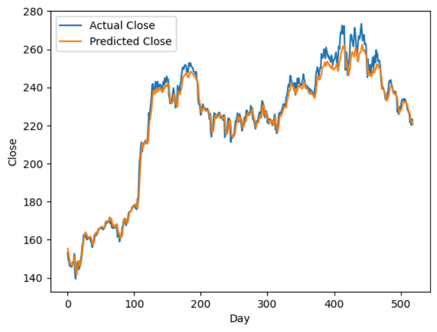




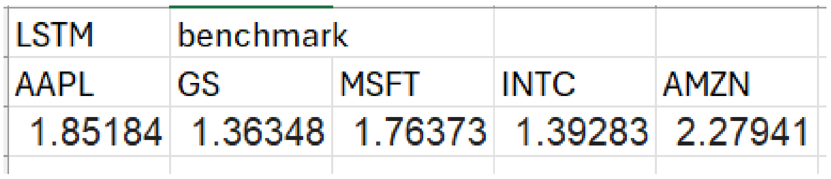




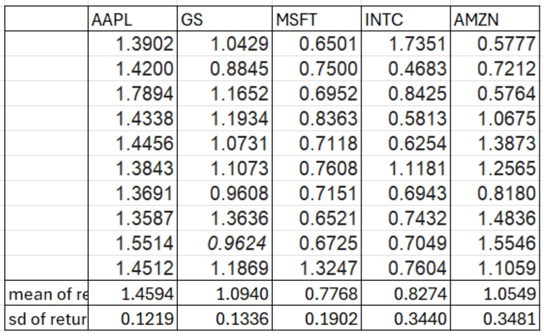
Since the predicted prices do not closely match the actual prices, we cannot utilize the trading strategy that involves comparing the next predicted price with the last closing price. This approach is viable only in scenarios where LSTM predictions exhibit high precision in testing data, which is occasionally observed. However, relying solely on this method would also entail an element of chance.



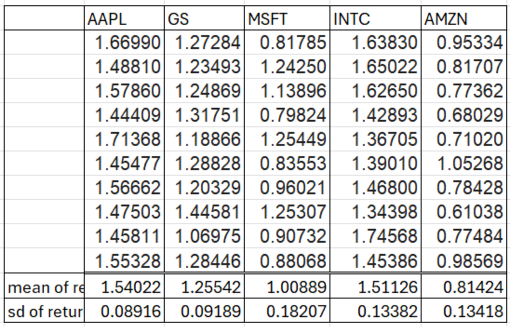
However, in the case of stocks such as AMZN and MSFT, which exhibit consistently strong performance without significant downturns, LSTM often produces unfavourable results. Consequently, we're left with the option of solely comparing the latest predicted price with the previous one to determine whether to take a long or short position. However, even with this approach, the outcomes tend to fluctuate significantly. Notably, only INTC demonstrates returns surpassing the benchmark on average. Moreover, each time the model is rerun with identical parameters and data, the strategy's return can vary by as much as 30%. To mitigate this variability, we adopt an averaging approach by generating 10 models and selecting the mean return. However, the inherent randomness of LSTM further complicates the development of trading strategies with consistent performance. The table below illustrates the extent of return variations.



LSTM	benchmark			
AAPL	GS	MSFT	INTC	AMZN
1.85184	1.36348	1.76373	1.39283	2.27941



	AAPL	GS	MSFT	INTC	AMZN
	1.3902	1.0429	0.6501	1.7351	0.5777
	1.4200	0.8845	0.7500	0.4683	0.7212
	1.7894	1.1652	0.6952	0.8425	0.5764
	1.4338	1.1934	0.8363	0.5813	1.0675
	1.4456	1.0731	0.7118	0.6254	1.3873
	1.3843	1.1073	0.7608	1.1181	1.2565
	1.3691	0.9608	0.7151	0.6943	0.8180
	1.3587	1.3636	0.6521	0.7432	1.4836
	1.5514	*0.9624*	0.6725	0.7049	1.5546
	1.4512	1.1869	1.3247	0.7604	1.1059
mean of re	1.4594	1.0940	0.7768	0.8274	1.0549
sd of retur	0.1219	0.1336	0.1902	0.3440	0.3481



	AAPL	GS	MSFT	INTC	AMZN
	1.66990	1.27284	0.81785	1.63830	0.95334
	1.48810	1.23493	1.24250	1.65022	0.81707
	1.57860	1.24869	1.13896	1.62650	0.77362
	1.44409	1.31751	0.79824	1.42893	0.68029
	1.71368	1.18866	1.25449	1.36705	0.71020
	1.45477	1.28828	0.83553	1.39010	1.05268
	1.56662	1.20329	0.96021	1.46800	0.78428
	1.47503	1.44581	1.25307	1.34398	0.61038
	1.45811	1.06975	0.90732	1.74568	0.77484
	1.55328	1.28446	0.88068	1.45386	0.98569
mean of re	1.54022	1.25542	1.00889	1.51126	0.81424
sd of retur	0.08916	0.09189	0.18207	0.13382	0.13418

<u>Strategy 2: ARIMA</u>

Stock prices often fluctuate sharply, which is caused by unexpected news or events. Such fluctuations are difficult for ARIMA models to predict. If the stock price were to surge due to other factors, it would be higher than the stock price predicted by the ARIMA model, so the ARIMA model will likely predict that the stock price will drop in the next period. On the contrary, if the stock price is plunged by other factors, the ARIMA model is likely to predict that the stock price will rise in the next period. However, if fluctuations in actual stock prices are trending by other factors, the ARIMA strategy exposes itself to the risk of losses due to opposite positions.

ARIMA prediction of an asset's future performance is based on past performance - ARIMA modelling can be inadequate for long-term forecasting because its past data is influenced by human factors. Thus, ARIMA should be used along with other tools to determine an asset's future performance.

ARIMA models are based on the assumption that the series is stationary or can be made stationary through differencing. This involves taking differences between consecutive observations to remove trends or seasonal effects. In ARIMA models, the order of integration (d parameter in (p, d, q)) specifies the number of times differencing is applied to the data to achieve stationarity. For non-stationary data, determining the correct order of differencing (the appropriate d value) can be challenging and subjective.

Analysis of optimization efforts and their impact on strategy performance

Strategy 1: LSTM

We tried to add various predictors, such as Exponential Moving Average, Relative Strength Index and Trading Volume, on top of the lag price. However, the performance did not improve. We infer that a larger number of predictors might be prone to overfitting, therefore we chose to employ a simpler model with fewer parameters. Thus, we just pick which lag closing price to include, which turns out to be 'close lag10', 'close lag7', 'close lag5', 'close lag3', 'close lag2', 'close lag1'.

However, when we try to predict the return instead of the closing price, RSI, EMA seem to be useful. Yet, as we observed that the models need to be greatly amended if we change the target variable to return, we did not proceed with it.
For example, we might consider standardizing the returns instead of normalizing them. Alternatively, we could opt to take no action on the target variable to preserve the direction (i.e., the sign) of the return. Additionally, we have tried to use Principal Component Analysis (PCA) for feature selection. However, the strategy performance did not show improvements. One possible reason is due to the limited features which makes the effect of PCA becomes less significant. Despite that, from PC1, we find out that the weight of RSI is quite significant, so RSI is an important factor in explaining the variance of the data.

Fine-tuning the hyper-parameter like the number of epochs and number of hidden layers indeed significantly improves the result of the model. From the backtesting result, the total return and standard deviation of return both improve. They will be discussed in the later part. LSTM is a type of deep learning which requires significant effort and knowledge to select the parameter.

Strategy 2: ARIMA

As written above, the ARIMA strategy has the disadvantage of being vulnerable to sudden changes in stock prices. The sudden change in stock prices can be seen as an increase in volatility in the rate of return of stock prices. Since each stock price will have its return volatility, we compared the 14-day average volatility and the 42-day average volatility of the return. So when short-term volatility was less than medium-term volatility, we used ARIMA's stock price forecast, and in other cases, we used other forecasts. Since the stock price may have fluctuated due to other external factors, the position calculated by ARIMA was not simply reversed, but Logistic Regression was used. The lagged price of 5 stocks, lagged trading volume of 5 stocks, volatility of stock returns, RSI of stock, volatility of the market, exchange rate, market index, and gold price were used as features of the logistic regression, and eight variables were selected using PCA to simplify the analysis.

Results of Backtesting and Performance Metrics

Strategy 1: LSTM

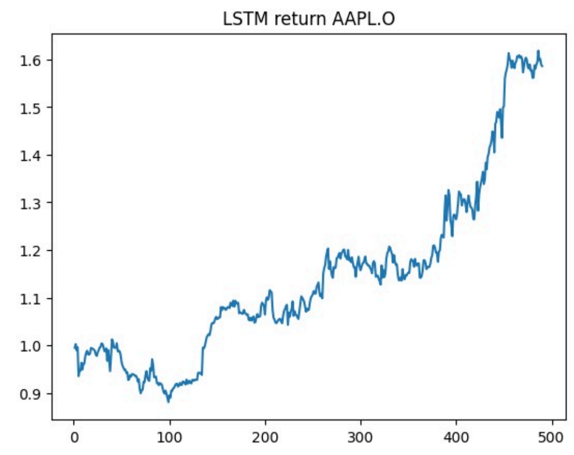
Mean Absolute Percentage Error (MAPE) is used to calculate percentage error and compare forecasts between the different data series. MAPE is calculated by the average of (Predicted Value - Actual Value) / Actual Value.

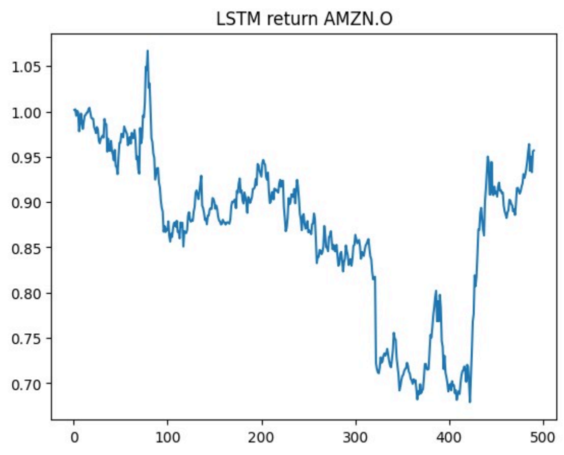
Before fine-tuning parameters:

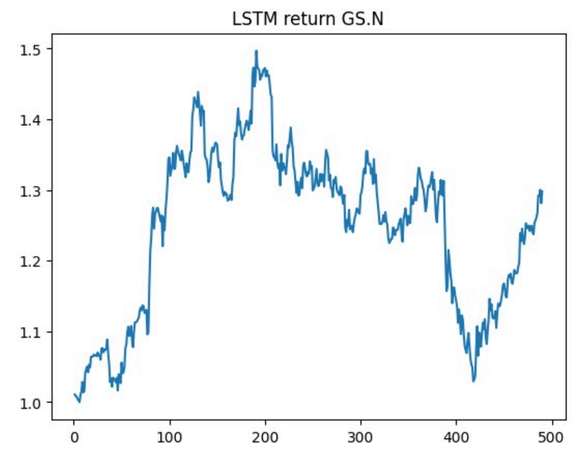
Stock	**Total return**		**s.d. of total return for 10 runs**	**MAPE**
	BM	LTSM (average of 10 runs)		
AAPL	1.8576	1.4594	0.1219	0.16153
AMZN	2.2794	1.0549	0.3481	0.52044
GS	1.3635	1.0940	0.1336	0.09887
INTC	1.3928	0.8274	0.3440	0.11676
MSFT	1.7637	0.7768	0.1902	0.19217

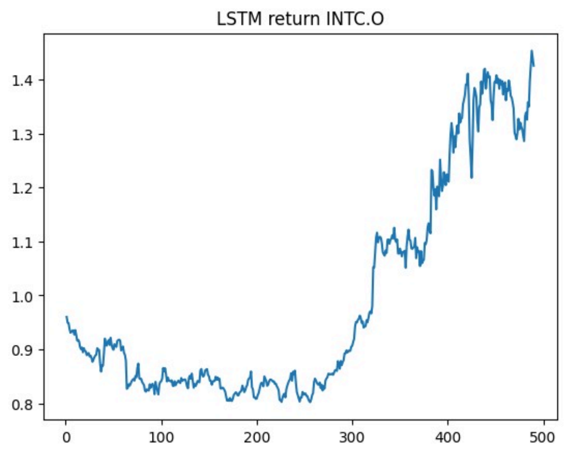
After fine-tuning parameters:

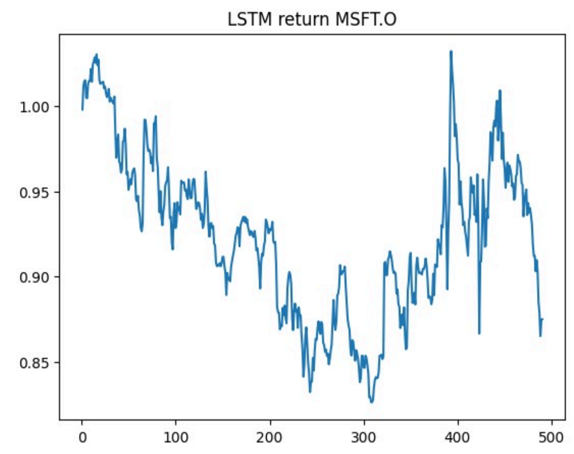
Stock	**Total of Return**		**s.d. of total return for 10 runs**	**MAPE**	**Accuracy of prediction (based on MAPE)**	**Accuracy (based on correct sign predicted)**
	BM	LTSM				
AAPL	1.8576	1.54022 (0.5402)	0.08916	0.10529 (10.5%)	89.5%	0.52061
AMZN	2.2794	0.81424 (-0.18576)	0.13418	0.34611 (34.6%)	65.4%	0.48755
GS	1.3635	1.25542 (0.2554)	0.09189	0.04629 (4.6%)	95.4%	0.49796
INTC	1.3928	1.51126 (0.51126)	0.13382	0.08627 (8.6%)	91.4%	0.51102
MSFT	1.7637	1.00889 (0.00889)	0.18207	0.25382 (25.3%)	74.7%	0.47816











We indeed show improvement after parameter fine-tuning. The total return all increased, except for AMZN. The standard deviation of the return of 10 runs also decreases which shows that the model converges and gives a more consistent output. Unfortunately, LSTM only beats BM in INTC and gives a relatively satisfying return in AAPL and GS. It suffers from loss in AMZN, which ironically is the stock with the best performance.

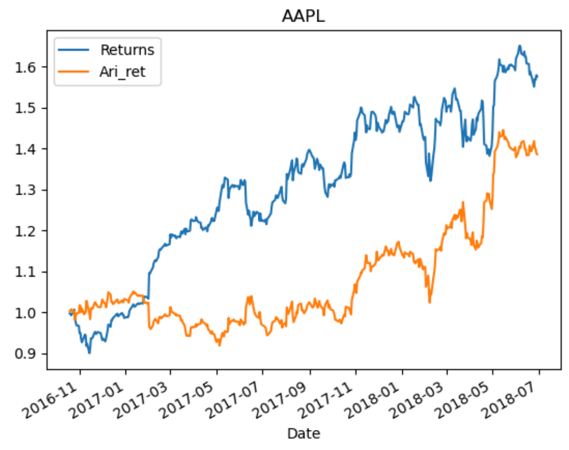
On the other hand, the accuracy based on the proportion of correct signs predicted, which is used for the trading strategy, is not so good. They are just around 50%. This also suggests why the returns given are not that ideal. Nonetheless, in some stocks like GS, even though the accuracy is slightly lower than 50%, the total return is 25%. It implies that the model may have identified some great rise and great drop, it just missed some point where the price change is not significant.

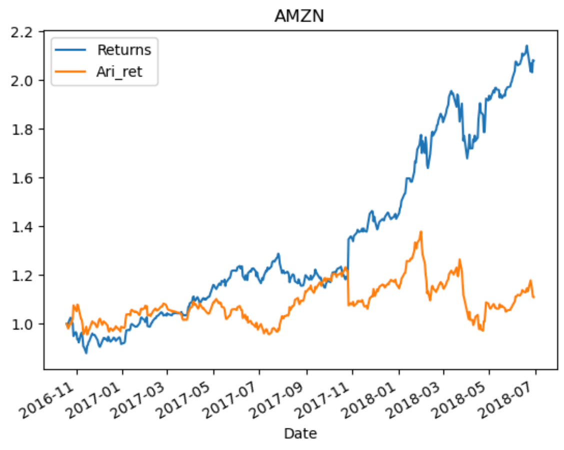
<u>Strategy 2: ARIMA</u>

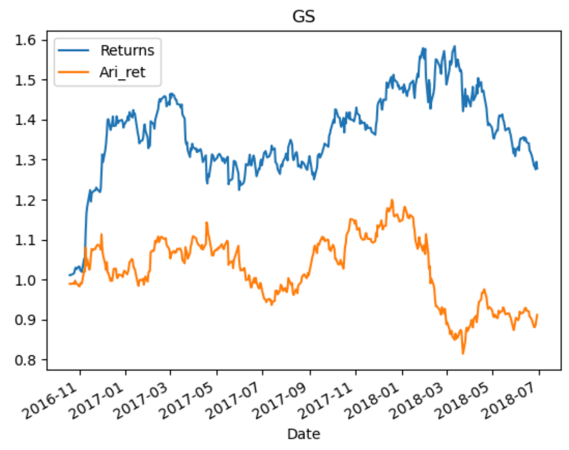
- Original ARIMA strategy

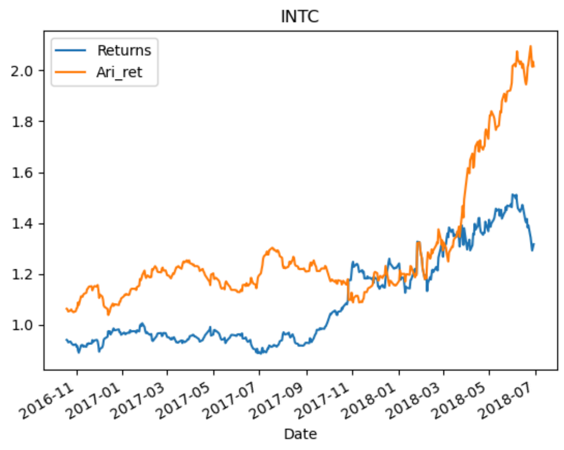
Stock	MAPE	Accuracy of Prediction
AAPL	17.6%	82.4%
AMZN	29.3%	70.7%
GS.N	8.9%	91.1%
INTC	17.8%	82.2%
MSFT	19.3%	80.7%

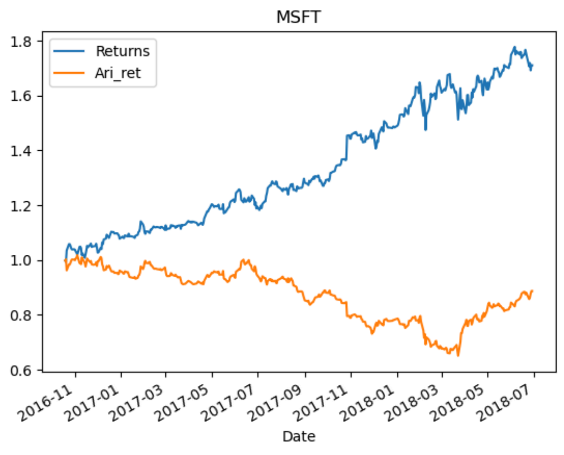
Comparison of total returns with benchmark strategy:











In the other four stocks except INTC, ARIMA does not beat BM. Furthermore, GS and MSFT suffered losses. Below is a table of total returns for each stock.

Stock	**Total return**	
	Benchmark (BM)	**ARIMA**
AAPL	1.576	1.386
AMZN	2.079	1.109
GS	1.278	0.911
INTC	1.317	2.016
MSFT	1.710	0.887

- Modified ARIMA

Stock	**Total return**		
	Benchmark (BM)	**ARIMA**	**Modified ARIMA**
AAPL	1.576	1.386	0.631
AMZN	2.079	1.109	1.590
GS	1.278	0.911	1.346
INTC	1.317	2.016	1.788
MSFT	1.710	0.887	1.289

Modified ARIMA outperforms the original ARIMA in four stocks and has also generated returns exceeding the benchmark return for GS.N and INTC.

Comparison of the Two Models

	LSTM		ARIMA (modified)	
	Accuracy (based on MAPE)	Total Return	Accuracy (based on MAPE)	Total Return
AAPL	89.5%	0.54022	82.4%	0.631
AMZN	65.4%	-0.18576	70.7%	1.590
GS	95.4%	0.2554	91.1%	1.346
INTC	91.4%	0.51126	82.2%	1.788
MSFT	74.7%	0.00889	80.7%	1.289
	Average accuracy: 83.28%		Average accuracy: 81.42%	

Modified ARIMA outperforms LSTM for all 5 stocks. However, LSTM has a higher average accuracy than that of modified ARIMA.

Future improvements

Strategy 1: LSTM

- Tailor-made model for each stock separately

We use the same parameter for all stock. However, as we can see after the fine-tuning, AMZN has a lower return. This may suggest that we need to tailor-made the parameter for each stock. We further suspect that for different timeframes, we may need to tune the model. Yet, this would give rise to the problem of overfitting. The model is not generic enough that it fits any timeframe and stock. Then, it is very risky to employ the model for trading in the real world as the financial market is ever-changing. Anything new to the model can happen.

- Using grid-search to find the best parameters and variables

The result may improve should we use a few nested for-loops to select parameters for prediction. However, the time complexity will become significantly high which requires extra optimization of the algorithm. For example, if the original time complexity is O(n), adding a new for loop to select the best parameter out of k choices will make the time complexity O(k*n). Yet, for every model, there are numerous parameters, say m parameters. Then the overall time complexity will become O(n*k^m) which will significantly reduce time efficiency.

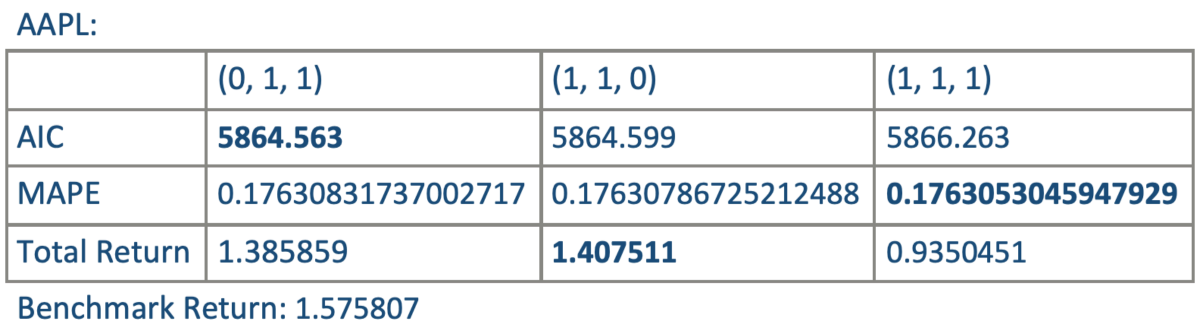
- Predict the signs of return instead of prices

Based on a preliminary LSTM model, predicting the direction gives a better total return. Yet, further investigation is needed to ensure correct logic is implemented. This may be because predicting direction is easier. After all, for the trading strategy, we just use the direction in the current model. We can also implement other variables like RSI, trading volume and EMA, and allow the neural network to employ feature selection implicitly because these variables are unable to improve the prediction of closing price doesn't imply that they are not useful in predicting the direction. It is suspected that short and long SMA may not be useful because LSTM can memorize short and long-term information. However, we can do some testing on it. All hyper-parameters will then adjust accordingly.

Strategy 2: ARIMA

- Further fine-tuning of parameters

In our ARIMA model, we have chosen the optimal parameters (p, d, q) based on their AIC. However, when we tested the data with other combinations of parameters, we noticed that the parameters with the lowest AIC might not have the highest total return and the lowest MAPE. This underscores the complexity of model selection and the importance of considering multiple evaluation metrics. Testing on a wider range of data and further fine-tuning may be required to obtain the optimal parameters.

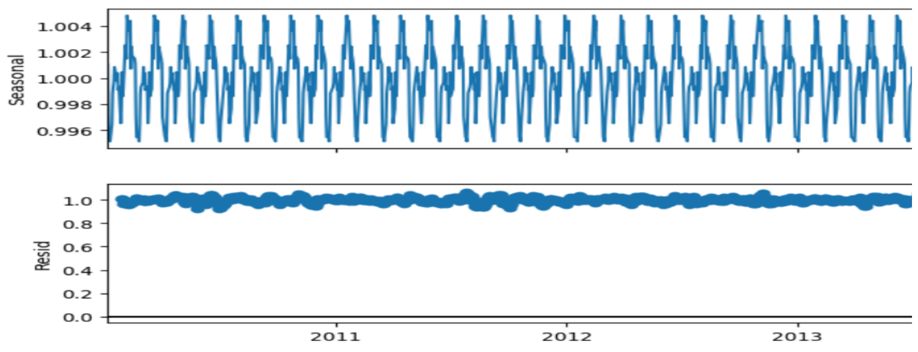

AAPL:

	(0, 1, 1)	(1, 1, 0)	(1, 1, 1)
AIC	**5864.563**	5864.599	5866.263
MAPE	0.17630831737002717	0.17630786725212488	**0.1763053045947929**
Total Return	1.385859	**1.407511**	0.9350451

Benchmark Return: 1.575807

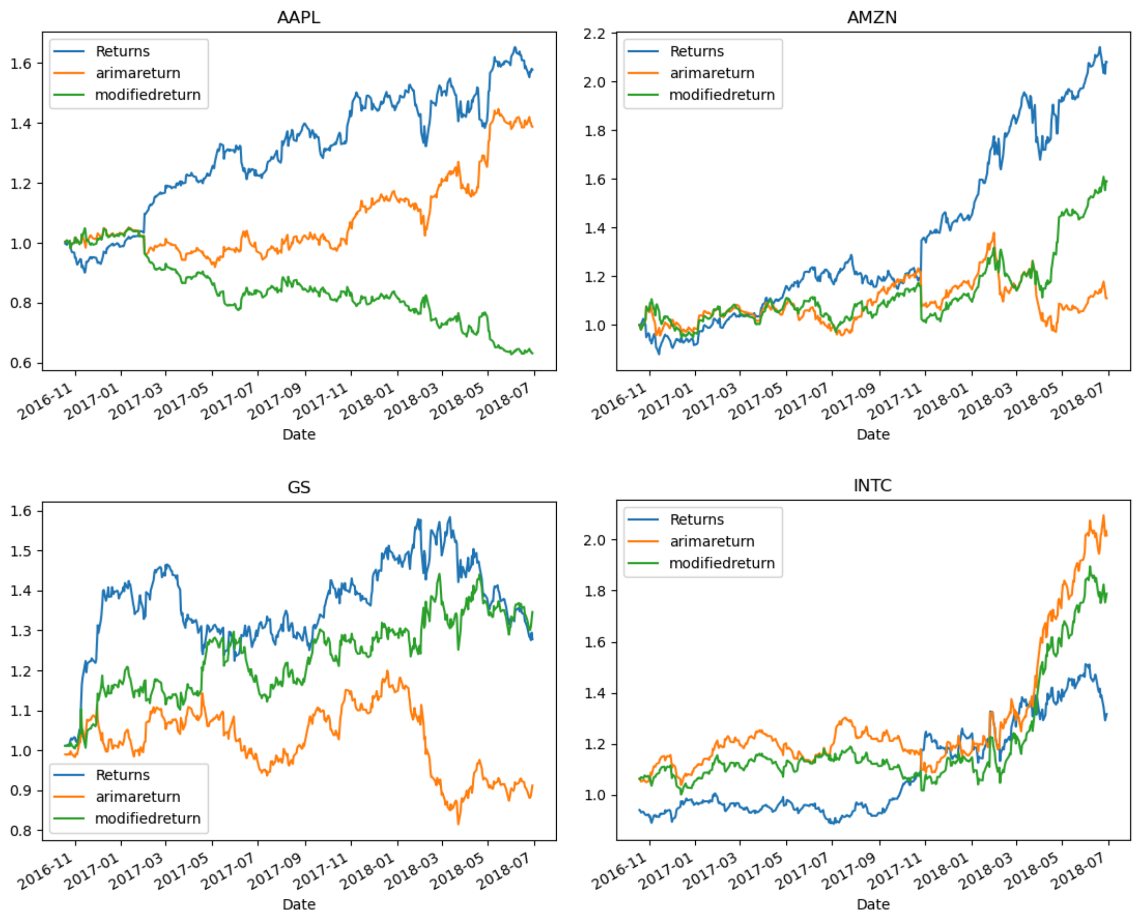
- Reducing the effect of seasonality

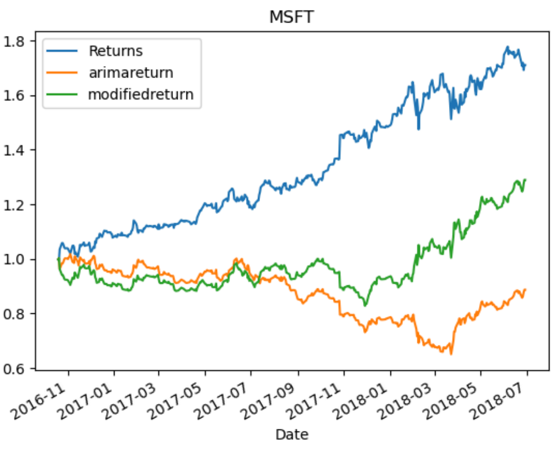
Stock prices exhibit clear signs of seasonality, revealing visible patterns over the course of each year. After decomposing the data into trend, seasonal, and residual components, it becomes evident that these temporal patterns are present. To improve the predictive capabilities of our models, it is important to integrate additional functions that account for this seasonality when forecasting stock prices. By incorporating these factors into our models, we can better capture and adapt to the cyclical fluctuations in the stock market, thereby improving the accuracy of our predictions.



- Use of combined strategy

To overcome ARIMA's vulnerability to sudden fluctuation in the market, we experimented with a combined strategy where Logistic Regression is used when short-term standard deviation exceeds medium-term standard deviation, which incorporated a variety of parameters to capture the change in market movement. The total return of this revised strategy can be shown by the green curve labelled as "modifiedreturn".



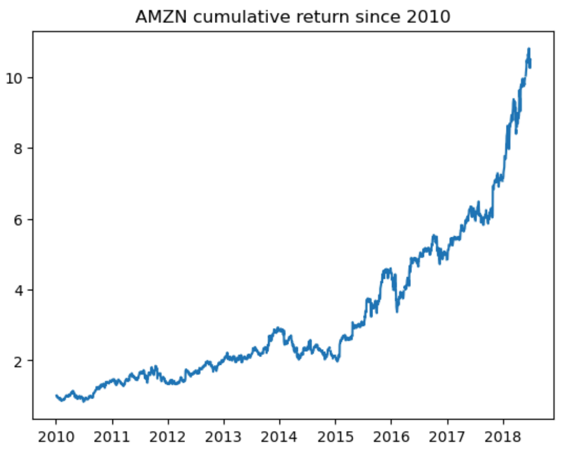


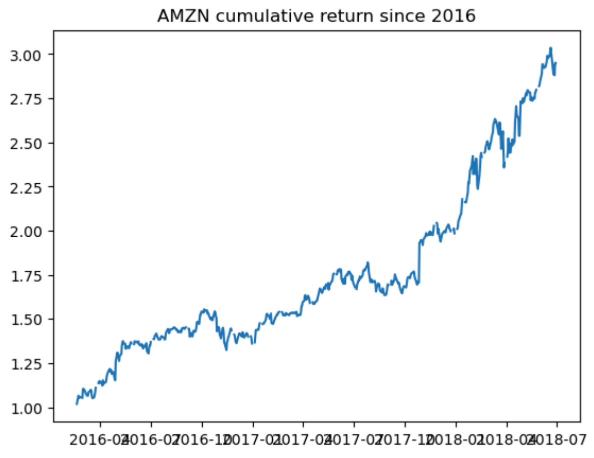
The graphs above show a general increase in return for AMZN, GS and MSFT compared to the original ARIMA strategy. For GS, the return of modified ARIMA beats the BM. In INTC, the rate of return was lowered but still beats BM. In AAPL, the return has been severely lowered. Below is a table of the total return for each stock.

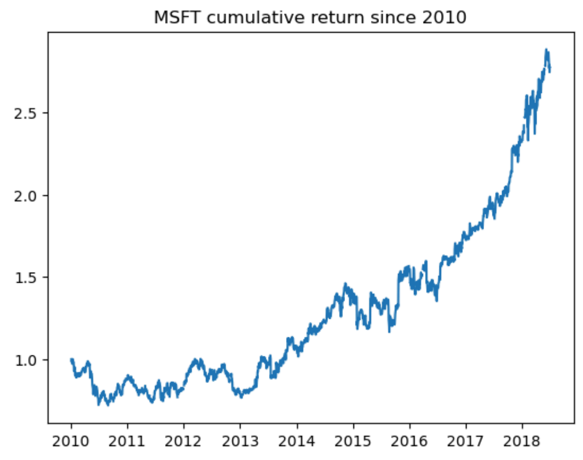
This result shows that although the revised strategy exhibits some enhancements in returns, its efficacy varies across the five stocks and throughout the time period. This suggests that for further improvement, we may need to refine our choice of parameters and explore other models for combinations.

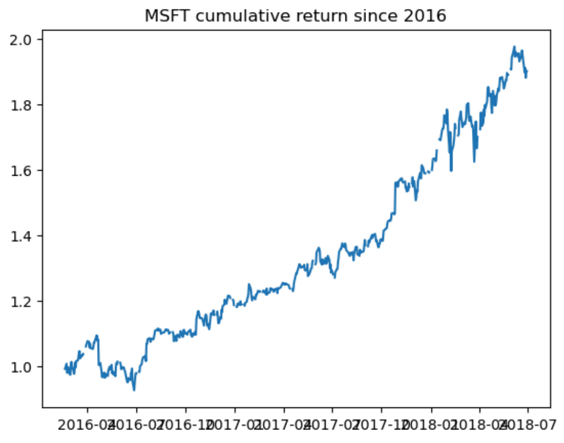
Reflection

AMZN.O has an astonishing benchmark result with no significant downward trend. Thus, it may be relatively challenging to outperform the benchmark strategy without further leverage. But that would be another story and the risk might be more significant. Therefore, the best strategy is just holding the stock for the entire period. Stocks in the tech industry like MSFT also have this issue.



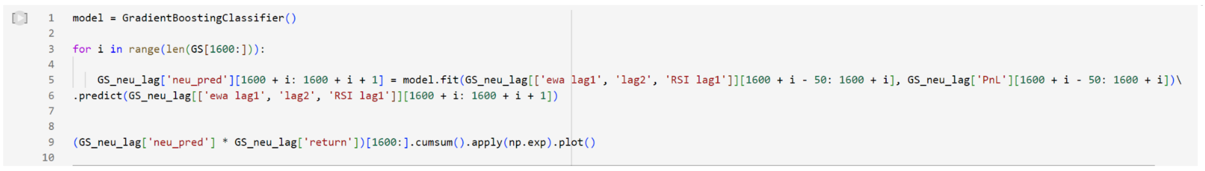






Another reflection is that we need to be more cautious when handling large numbers of data. For example, when we accidentally include “today” data as one of the predictors, the model shows consistently high performance on the testing data, even if other predictors are lagged data which are correctly input and only one foreseeing variable is used.

Time series data have some kind of trends and show different features from time to time. This is also the challenging part of predicting stock prices. In response, we used the idea of moving windows. For each day, we make a new ML model using data from the previous 50 days (the number of training data can be adjusted) and use this model to predict a single day right after the period of the data.



```
model = GradientBoostingClassifier()

for i in range(len(GS[1600:])):

    GS_neu_lag['neu_pred'][1600 + i: 1600 + i + 1] = model.fit(GS_neu_lag[['ewa lag1', 'lag2', 'RSI lag1']][1600 + i - 50: 1600 + i], GS_neu_lag['PnL'][1600 + i - 50: 1600 + i])\
.predict(GS_neu_lag[['ewa lag1', 'lag2', 'RSI lag1']][1600 + i: 1600 + i + 1])


(GS_neu_lag['neu_pred'] * GS_neu_lag['return'])[1600:].cumsum().apply(np.exp).plot()

```

Although this method generally performs better than LSTM, the return is still below the benchmark strategy. As we want to focus more on time series analysis, we employ LSTM in this project. In future research, we may proceed with using classification ML models such as Logistic Regression and MLPClassifier to predict the direction of stock price movements which may be easier than predicting stock prices.

Using ML models to predict stock prices is indeed challenging. Therefore, even with tools like Python and machine learning algorithms, generating consistent profit remains difficult. A model that proves profitable for one stock may not apply to other assets. Furthermore, even if we apply the model to the same asset, it may give a disappointing result in a new time frame and cannot overcome a new financial incident like the COVID pandemic and interest rate hike. This underscores the significance of expertise in quantitative analysis and investment within financial institutions, where a higher level of knowledge and skill is often required. Nevertheless, the pursuit of understanding and improving trading strategies is always an interesting topic to investigate. It is hoped that with more knowledge acquired, we can formulate a better trading strategy in the future.

References

https://medium.com/shikhars-data-science-projects/predicting-stock-prices-using-arima-fourier-transformation-and-deep-learning-e5fb4f693c85

https://www.analyticsvidhya.com/blog/2021/07/stock-market-forecasting-using-time-series-analysis-with-arima-model/

https://www.kaggle.com/code/nageshsingh/stock-market-forecasting-arima/notebook

https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

https://www.sciencedirect.com/science/article/pii/S2666827022000378

Appendix

Failure / testing code folder includes code for some preliminary testing on different models and some plotting such as plotting correlation matrix, testing on using for-loop to generate a new model for each new day, and testing on using more variables in LSTM to predict return.