| Problem Chosen | 2022 MCM/ICM Summary Sheet | Team Control Number |
|---|---|---|
| C | | 2221952 |

# Maximizing Investment Profits via a Comprehensive Voting Model


**Summary**

Quantitative trading is an advanced techniques that excavates insights from vast historical data and integrate multiple probability models to formulate the optimal strategies. In this project, we constructed a quantitative strategy model that combines financial and mathematical analysis, and designed a voting mechanism to improve the quality of decisions.

In question 1, we built up a prediction model based on external training data by multiple approaches. Firstly, we obtain the "buy" "hold" and "sell" signal from the **Moving Average Convergence Divergence (MACD)** strategy, a widely-applied financial analytic tool. Then, the **Bull-bear market value** quantifies the investment value based on the judgement of tendency and volatility, which provides an important criterion on value scoring in following steps. To achieve accurate predictions, we found hyperparameters of **LSTM** and pretrained it on our training set. We applied **ARIMA(p,1,q)** that refreshes its coefficients according to updated new data. We designed an **online learning** scheme so that the LSTM and ARIMA model can update their best predictors throughout their learning process. A scoring function is built up to evaluate the worthiness to invest, and we will buy or sell the amount of Bitcoin or gold proportional to the exponent of this score.

In question 2, we proved the optimality of our model in two ways. Firstly, the parameters chosen for each sub-models has helped the predictions attain the minimum mean-squared error. The second way is showing that it attains a local maximum profit. We added a Gaussian(0,1) disturbance on the transaction amounts, and find out that 95% of the corresponding strategies result in a profit lower than ours.

In question 3, we showed that higher transaction cost decreases transaction frequency. Although our trading strategy depends on transcation cost $\alpha$, our model is stable to small perturbation of transcation rate.

In the end, we improves the robustness of our model by replacing the original hierarchical decision process with a new **voting system**. The MACD signal is no longer the decisive term to the investment direction, but become a cooperator with the scoring system to improve the precision. The improved model avoids the randomness of mistakes resulted from the MACD model, and our strategy helps the trader get a portfolio that worths $78417.14 at Sept., 2016.



**Keywords**: Autoregressive Integrated Moving Average (ARIMA), Long-short term memory (LSTM), online learning, sensitivity analysis, multi-model voting

# Contents

# 1 Introduction

## 1.1 Restatement of the Problem

Bitcoins and gold are two hot topics in the investment market. The former have gone through huge oscillations in previous years, and the later is usually the symbol of the value-preserving asset. In this problem, we made a portfolio consisting of only cash and the two assets and developed a strategic framework to maximize the return of the investment on this portfolio.

Starting with \$1000 on 9/11/2016, to reach the best return by 9/10/2021. Bitcoin can be traded every day, but gold can only be traded if the market is open. The commission for transaction (purchase or sale) of bitcoins and gold costs $\alpha_{bitcoin} = 2\%$ and $\alpha_{gold} = 1\%$, respectively. No cost is required to hold an asset. Our tasks can be divided into the following steps:

- Develop a model that gives the best daily trading strategy based only on price data up to that day. The "best" can be interpreted as maximum return on investment (ROI) or the stability of return towards the market risks.
- Show that our model reaches the best result. To finish this requirement, we can set up an objective function or evaluation criteria, and show that our strategy achieves the optimal value.
- Find the sensitivity of the strategy to transaction costs, and how do transaction costs affect the strategy and results
- Communicate the strategy, model, and results to the trader in a memorandum.

## 1.2 Our Work

To solve this problem, we worked out our model with the following steps:

- Built the price forecasting tools based on the training dataset, which show high accuracy on the test set.
- Combined the prediction on price, trends and risks, we developed a scoring system that assesses whether it is worthy to have a position on each asset. Using the score, we designed trade score formula that defines how much amount we should trade.
- To eliminate the mistake brought by the error of some sub-model, we designed a voting decision-making flow that weighs the financial and analytical predictions.

Our model helps the traders obtain the local maximum ROI, and the strategy subjects to change if the transaction cost varies.

### 1.3 Training and testing dataset

Some sub-models like MACD and bull market value returns the feedback spontaneously with the current market and therefore can be directly applied on the test set. However, the other sub-models need to be trained on other datasets. To satisfy this requirement, we downloaded the historic price from 20014 to 2016 as the training set and made that from 2016 to 2021 the test set.

### 1.4 Assumptions and Justifications

When discussing the optimal strategies of transactions based on the market situation, we usually need to adjust our actions with the ever-changing market situation. In this problem, the data is provided on daily basis and our models are expected to be focused on the official data. To simplify our analysis and facilitate the data-driven modeling, we raised several assumptions in this section.

- **The basic market assumptions hold throughout the time.** There is no arbitrage in the market, and we assume no external events with significant influence on the gold's or bitcoin's prices will occur within the period time of our discussion. Also, traders do not know the future prices.
- **Traders only perform one action at most on each kind of asset per day.** That means, we choose either to buy or to sell a certain amount of asset in a single day, and the action must complete in one go. No hedging or shorting is allowed in our discussion.
- **The change in asset prices is slight on days the market is closed**. Even if we can fit the existing data to complete the blank values on non-trading days, we assume that the gold price on weekends is unchanged, as the price is decided by the demand and supply and there are no buyers and sellers of gold on weekends.

## 2 Notations

| Symbols | Definitions |
|---|---|
| $\text{EMA}_1$ | Short-term exponential moving average |
| $\text{EMA}_2$ | Long-term exponential moving average |
| $r_t$ | Daily return on day $t$ |
| $e_t$ | Residual of day $t$ |
| $B_{price}(t)$ | The value of bitcoin price on $t$ days from 2016/9/11 |
| $G_{price}(t)$ | The value of gold price on $t$ days from 2016/9/11 |
| Subscript $b$ | Represents the word "bitcoin" |
| Subscript $g$ | Represents the word "gold" |

# 3 Question 1: Model development

We used the integration of financial and machine learning models to predict the opportunity of transactions. Among several popular stock trading techniques, we chose Moving Average Convergence Divergence (MACD) as the financial signal, as it is both stable in value and sensitive in the market trend.

We also applied neural network and time series model to predict the asset prices, as it provides feedback from a technical perspective. Noticing that there are 10 nans in the gold price, we filled them using cubic spline.

## 3.1 The MACD strategy

Moving Average Convergence Divergence (MACD) is used to detect the signal to trade in or trade out. It shows the best return and profit in a lot of trading scenarios, as it combines the long-term and short-term moving 19 trend [1]. The first step is to compute the Exponential Moving Average curve, which is defined by

$$\text{EMA}(N, x_n) = \frac{2x_n + (N-1)\text{EMA}(N, x_{n-1})}{N+1}$$

We computed both short-term EMA where $N = 12$ and long-term EMA where $N = 26$. The Diff is calculated as $\text{Diff} = \text{EMA}_1 - \text{EMA}_2$ and DEA is the 12-day moving average $\text{DEA} = \frac{1}{12}\sum_{k=1}^{12} \text{Diff}_k$. Then, we can calculate the MACD from the equation

$$\text{MACD} = 3 \times (\text{Diff} - \text{DEA}).$$

The visualization shows that the two EMA lines are very close to the prices. The MACD, Diff, and DEA can be both negative and positive, as they provide signals of the transformation of the market. If the MACD of one day is larger than zero, the MACD of the day before yesterday is smaller than zero. It is an indicator that the price of gold tends to rise. When the MACD is smaller than zero, and the MACD was larger than zero the day before yesterday, then it is the signal to sell the gold.

We back-tested the MACD method on the training data to trade bitcoin and gold respectively, resulting in 20% to 50% cumulative profits in both cases. The testing result shows that the MACD is feasible, but we need to take the budget and risk capability in the next step.

## 3.2 The signal of a bull market or bear market

We first define the percentage gain as $G_{gain}(i)$, which is calculated as

$$G_{gain}(i) = G_{price}(i) - G_{price}(i-1),$$

$$B_{gain}(i) = B_{price}(i) - B_{price}(i-1).$$

We introduced the discount weighted mean of $G_{gain}$ from day $i-n$ to day $i-1$, by

$$\bar{G}_{gain}(i, n) = \frac{1}{\sum_{k=i-n}^{i-1}(0.95)^k} \sum_{k=i-n}^{i-1} (0.95)^k \times G_{price}(k),$$

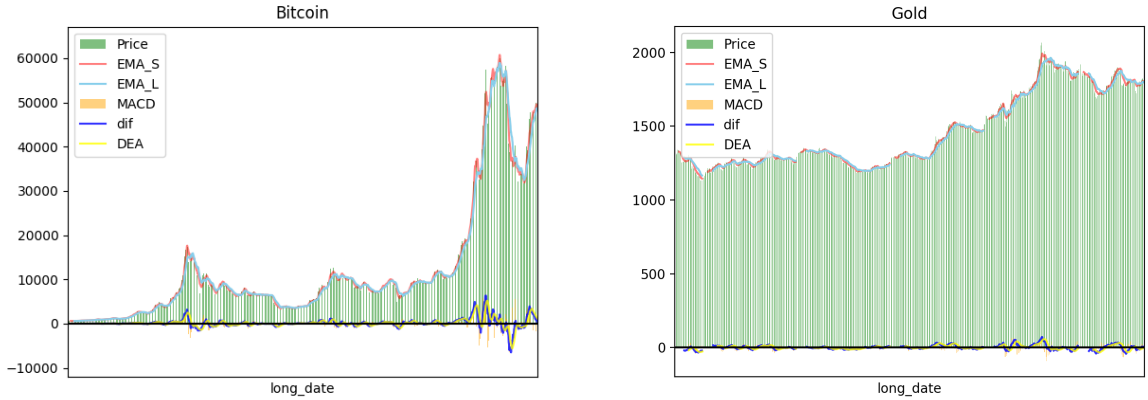


Figure 1: The MACD lines of bitcoins and gold

$$\bar{B}_{gain}(i,n) = \frac{1}{\sum_{k=i-n}^{i-1}(0.95)^k} \sum_{k=i-n}^{i-1} (0.95)^k \times B_{price}(k).$$

Here $n$ is chosen to balance the stability of data and the tendency of the volatility. Figure 3 and figure 2 shows the discount weighted mean with respect to date.

We choose $n = 20$ for $\bar{G}_{gain}$, since the volatility fits well and the data is stable. For similar reasons, we choose $n = 5$ for $\bar{B}_{gain}$.

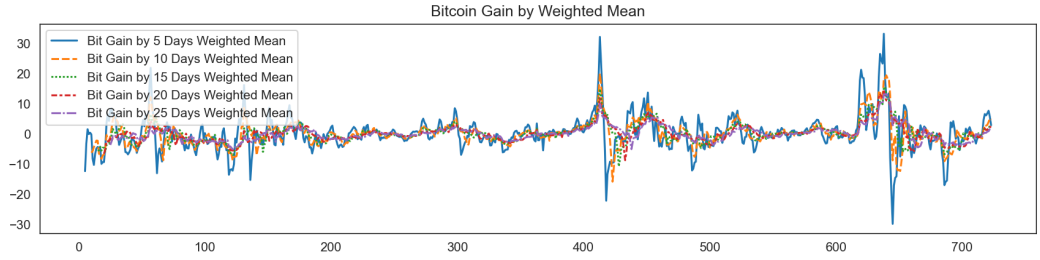


Figure 2: $\bar{B}_{gain}$ average on training set

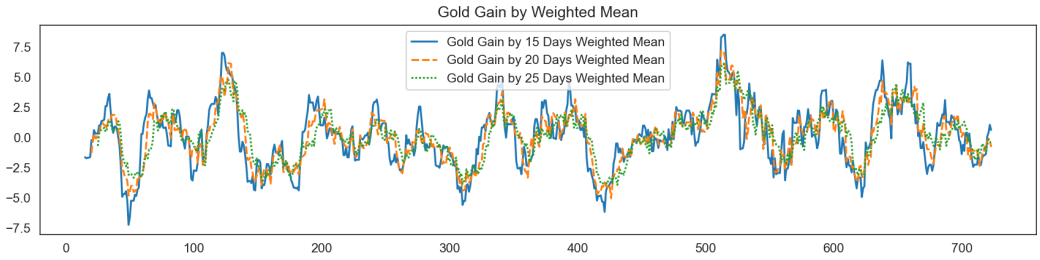


Figure 3: $\bar{G}_{gain}$ average on training set

### The Bull Market Value (BMV)

The Bull Market Value [2] is used to judge the tendency of the volatility of the price. The definitions are ($w = 0.42$):

$$\text{BMV}_b(i) = w \times \sum_{k=i-5}^{i-1} B_{gain}(k) + (1-w) \times \sum_{k=i-5}^{i-1} \bar{B}_{gain}(k,5)$$

$$\text{BMV}_g(i) = w \times \sum_{k=i-15}^{i-1} G_{gain}(k) + (1-w) \times \sum_{k=i-15}^{i-1} \bar{G}_{gain}(k,15)$$

For bitcoins, if $\text{BWV}_b(\text{i}) \geq \text{mean}_i(\text{BMV}_b)$, then the $i$th day is regarded as bull market. If $\text{BWV}_b(i) < \text{mean}_i(\text{BMV}_b)$, then the $i$th day is regarded as bear market. Similarly for gold. Figure 4 below shows the judgement of each day.

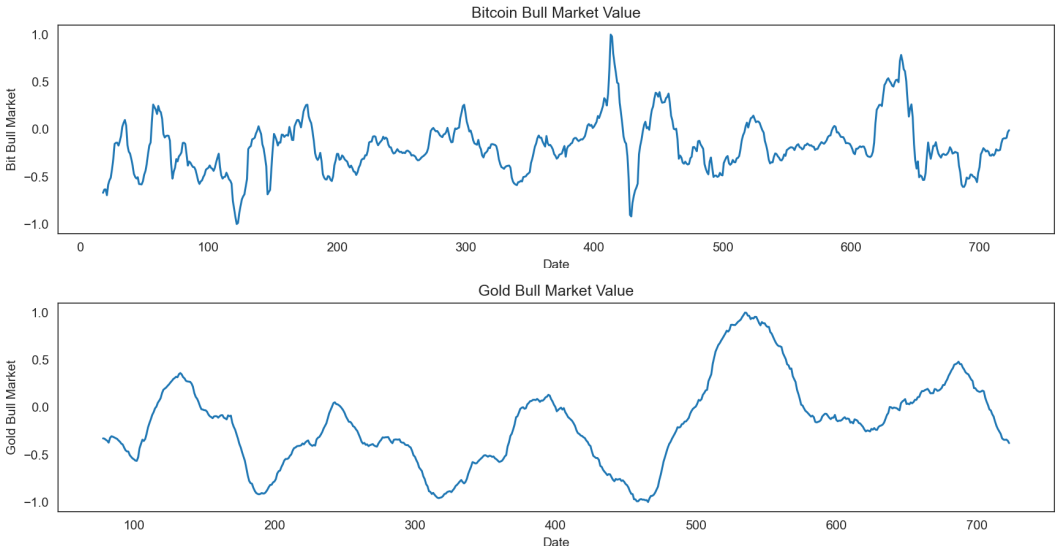


Figure 4: The bull and bear market of bitcoin and gold

### The risk evaluation of bitcoin and gold

The risk [3] is an important judgement for the strategy. We formulated the evaluation of the gold's and bitcoin's risk as

$$\text{Risk}_g(i) = 1 - w \times \text{BMV}_g(i) + (1-w) \times \bar{G}_{gain}(i,20),$$

$$\text{Risk}_b(i) = 1 - w \times \text{BMV}_b(i) + (1-w) \times \bar{B}_{gain}(i,5),$$

where $w = 0.42$. Figure 6 describes the risks of bitcoin and gold from 2016/9/11 to 2021/9/10. Using the formula above, we can calculate our prediction of risk at any date based on the inputted data. The risk will be taken into account in the assessment of trading values.

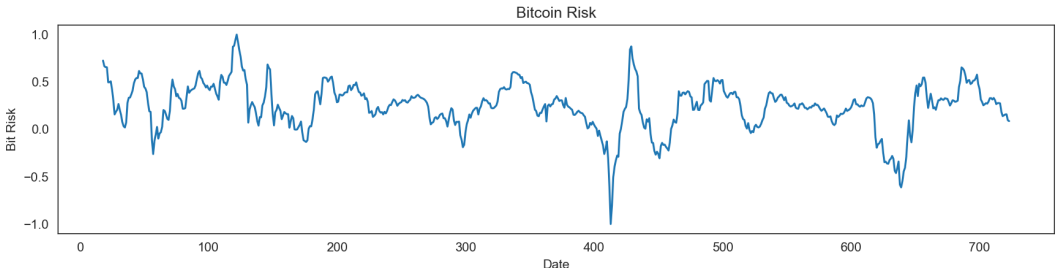


Figure 5: The risk of bitcoin

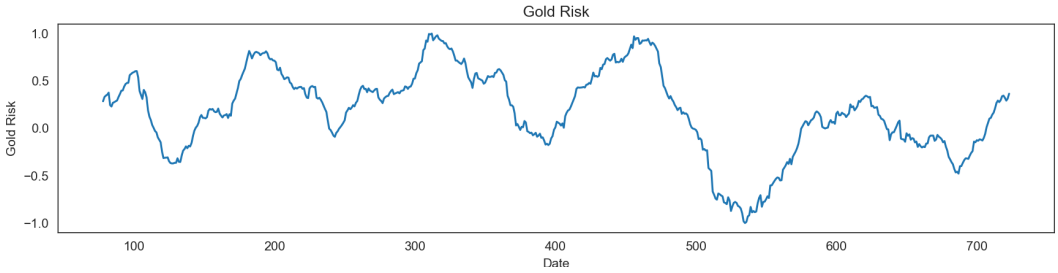


Figure 6: The risk of gold

## 3.3 Gray prediction for price prediction

Gray prediction is a naive model that is efficient for short-term prediction. This model does not need to be trained. Hence, we directly used this model on test set. From the 11th day, we predict the price of the following day by the historic price. The result of grey prediction is shown in figure 7.

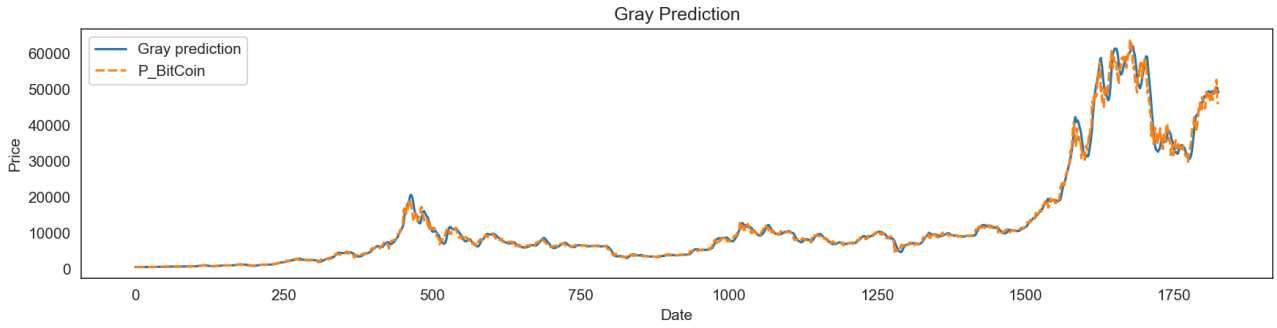


Figure 7: Gray prediction model

We used root mean squared error (RMSE) to determine the accuracy of the model, which is defined as

$$\text{RMSE}(x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

The result shows that RMSE $= 452.58$.

## 3.4 ARIMA for price prediction

Time series phenomenon is usually discovered in the financial market, and we applied the Autoregressive Integrated Moving Average (ARIMA) for the prediction. The first step is to test whether the time series of prices are applicable for time series modeling. The Augmented Dickey-Fuller (ADF) test reveals whether a series is stable. We found that the prices of both bitcoin and gold are not stationary through the ADF test. However, the 1st order difference of the two series are both stable, meaning that we can fit the ARIMA(p,1,q) model to them.

| p-value | | ADF Test<br>$H_0$: Not stationary series |
|---|---|---|
| Bitcoin | Original | 0.9000 |
| | $1^{st}$ diff | $< 0.01^{**}$ |
| Gold | Original | 0.9217 |
| | $1^{st}$ diff | $< 0.01^{**}$ |

Table 2: ADF test of the price data

Having determined the differentiation order, we fitted the model with the training data. Starting from $l_0 = 6$, we took the first $l$ samples to fit the ARIMA(p,0,q) model by comparing AIC values and extending one unit of the data length to re-fit the model. The iteration is summarized in Algorithm 1.

**Algorithm 1** ARIMA iteration fitting

**Require:** Historical data of asset price
  Initialize: $d = 1$
  **for** Every transaction day **do**
    Fit ARIMA(p,d,q) on the historical data
    Append today's actual data to the historical data
  **end for**
**Ensure:** The final ARIMA(p,d,q) model

The algorithm has the RMSE for bitcoin and gold, respectively, 828.89 and 13.96 on the test sets.

## 3.5 LSTM for price prediction

Long Short-Term Memory (LSTM) can grasp the dynamic structure of price over time. With gated memory cells, it attains high prediction capacity. We built a modified LSTM that reads 20-days-long price sequences and predicts the prices of the following 3 days. We added the input price to the output of LSTM because we assumed tomorrow's price highly depends on today's. We used training data of the stock market before 9/11/2016 to train the LSTM and got a source model. The loss function is defined as the mean squared error, which converges very quickly in the 500-epoch training.

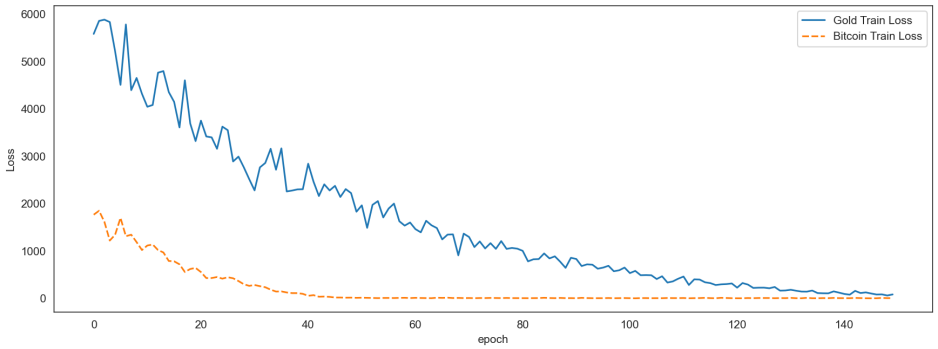


Figure 8: The loss of LSTM model

Then we turned this source model based on the historical trade price of bitcoin and gold. We found that the best LSTM should have 1 hidden layer and the hidden size should be 64. The models are used to predict the prices in September, 2016 to September, 2021 period, obtaining the RMSE $= 714$ for bitcoin and RMSE $= 19.72$ for gold.

We use the data of the last 20 days to predict the price of the next day. Day 1 prediction is for the next day and day 2 prediction is for the day after tomorrow.

## 3.6 Investment strategy model

Having built the prediction models above, we integrated them as a decision model. The algorithm is as follows:

**Algorithm 2** Investment Algorithm

**Require:** $k_{trade}, w_{price}, w_{bull}, w_{risk}$, are pre-determined parameters
  **for** each day **do**
    Obtain the trade signal (buy/sell) from MACD model based on the past 60 days of bitcoin and gold price
    **if** signal = trade **then**
      Obtain the predicted gain $Value_{gain}$ from LSTM and ARIMA models
      Compute the bull market values $Value_{bull}$ and risks $Value_{risk}$.
      Compute $TS =$ Normalize$(w_{price} \times Value_{price} + w_{bull} \times Value_{bull} - w_{risk} \times Value_{risk})$.

      **if** signal = buy **then**
        Compute the buy amount $n_{buy} = k_{trade} \times e^{TS}$
      **else**
        Compute the sell amount $n_{sell} = k_{trade} \times e^{-TS}$
      **end if**
      Trade
    **end if**
  **end for**

To build the strategy model, we need to consider the risk, the predicted profit ($D_{bitcoin}$). The trade score (TS) is used to quantify whether an asset is worthy to buy which is defined as

$$\text{TS}_b(i) = \text{Normalize}(w_{price_b} \times D_b + w_{\text{BMV}_b} \times \text{BMV}_b - w_{\text{risk}_b} \times \text{risk}_b)$$
$$\text{TS}_g(i) = \text{Normalize}(w_{price_g} \times D_g + w_{\text{BMV}_g} \times \text{BMV}_g - w_{\text{risk}_g} \times \text{risk}_g)$$

where $w$ represents the weight. TS is used to determine the transaction volumes. In day $t$, if we have decided to buy bitcoins, then the amount we buy is computed by

$$n_{\text{buy},b} = k_b \times e^{\text{TS}_b}$$

If we decide to buy gold in day $t$, the amount is computed by

$$n_{\text{buy},g} = k_g \times e^{\text{TS}_g}$$

If we decide to sell bitcoin in day $t$, the amount is computed by

$$n_{\text{sell},b} = k_b \times e^{-\text{TS}_b}$$

If we decide to sell gold in day $t$, the amount is computed by

$$n_{\text{sell},g} = k_g \times e^{-\text{TS}_g}$$

where $k$ is the hyperparameter related to transaction amount and $w_{price}, w_{BMV}, w_{risk}$ are weights in computing TS.

## 3.7 Choosing hyperparameters

We chose hyperparameters on the training set whose date is from 9/17/2014 to 9/9/2016. We wanted the TS be high when the price is increasing. We set $w_{price}$ to be 1 and changed $w_{BMV}, w_{risk}$ to fit the TS curve. The result is as follows:

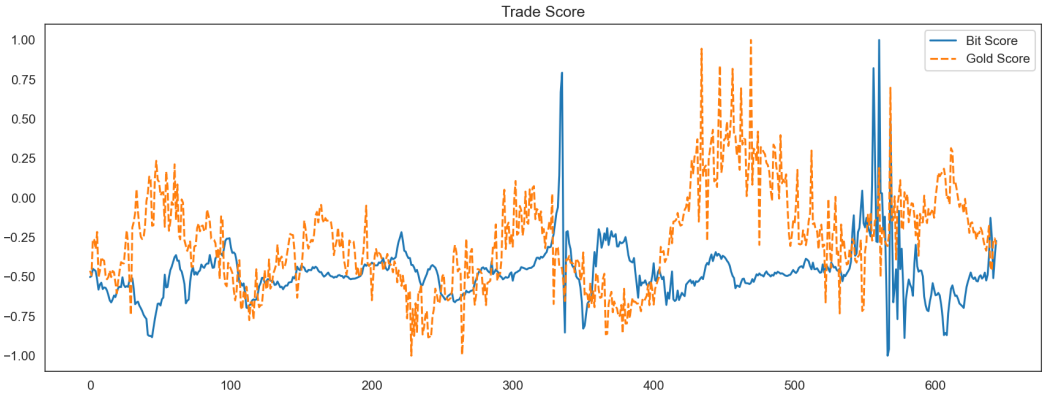


Figure 9: Total Score

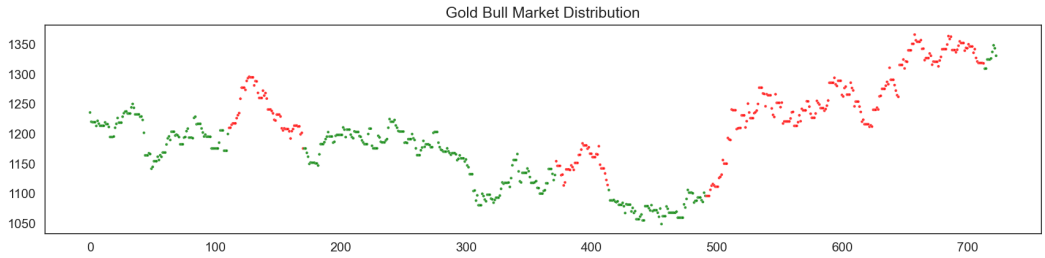


Figure 10: Gold Bull Market

Comparing two figures above, we can see that when TS is large, the price trend to increase.

For each transaction, we wanted our strategy to balance the return and risk. Hence, each time when there is a signal to trade, it spends about 10% of the remaining cash. The result is as follows:

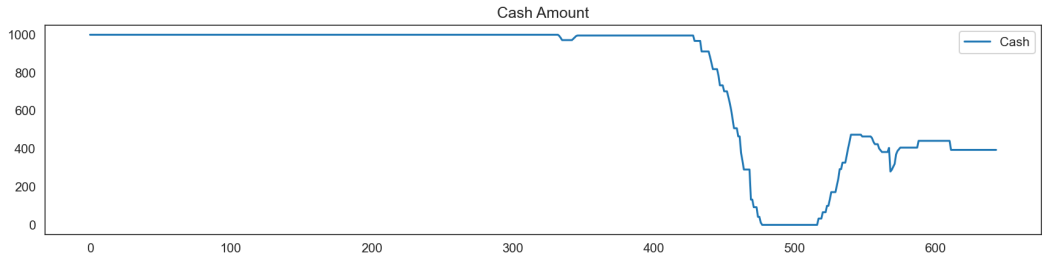


Figure 11: Cash Amount

Finally, the hyperparameters we chose are

| | $w_{price}$ | $w_{bull}$ | $w_{risk}$ | $k_{trade}$ |
|---|---|---|---|---|
| bitcoin | 1 | 5 | 10 | 54 |
| gold | 5 | 1 | 12 | 124 |

### 3.8 The result of the model

We test our model on the test data whose date is from 9/11/2016 to 9/10/2021. The result of the model shows that our investment portfolio deserves **$30104** in 9/10/2021. However, after we improved the model, our investment portfolio finally reaches to **$78417** (methods for improvement to be discussed in the later section).

## 4 Question 2: Evidence of optimality

Consider the model after improvement. To prove that our strategy is optimal, in each transaction step, we gave our transaction amount some randomness and repeat the whole

transaction process from 2016 to 2016 for 1000 times. If the profit of 95% of the random experiment is lower than our strategy, then it is persuasive to prove that our strategy is at least a local maximum. The reason is that in reality, we tolerate some randomness to improve the profit, but the probability for such improvement should be low.

We introduced noises to the price series and tested the model. For each transaction, we give our transaction amount Gaussian noise with $\mu = 1, \sigma^2 = 0.2$ (i.e. $n \implies n \times \mathcal{N}(1, 0.2)$). Also, in each transaction, it has the probability of 0.1 to reverse the action (i.e. sell to buy and vice versa). The result shows that **95.2%** of the experiments have results that is lower than our return (red line), which means our model is the optimal.

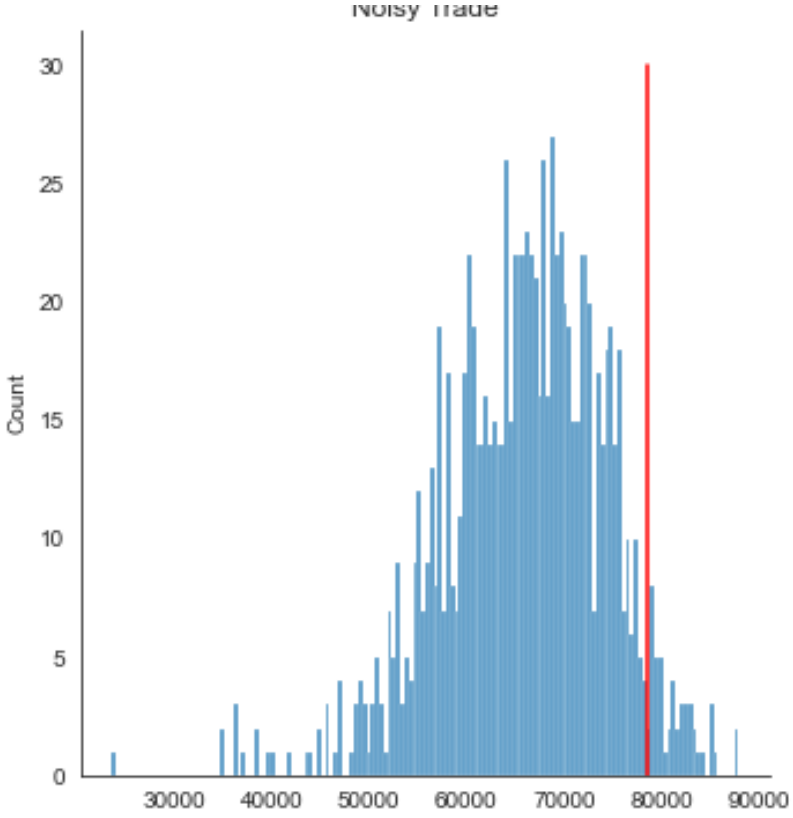


Figure 12: Noise Trade

# 5 Question 3: Sensitivity analysis of transaction cost

Under the same market condition, the variety in transaction costs may affect the decision. The MACD method, as a part of the integrated model, raises signals regardless of transaction costs. If we only uses MACD to make decisions, the return will be hugely canceled by transaction costs if the costs go higher. Therefore, a combination is needed to comprehensively evaluate the feasibility of each transaction.

In this problem's setting, transaction costs of Bitcoin and gold are 2% and 1%, respectively, which are related large in the comparison of stock market. We must ensure that

each transaction gets a good reward. Hence, we also compared trade scores with transaction cost. The higher transaction cost, the higher trade scores are needed to activate a transaction. In most cases, trade scores are much higher than transaction cost. Thus, our strategy is not sensitive to small perturbation of transaction cost.

In out model, transaction cost only affects the frequency of transaction. If the transaction cost decreases and the corresponding weight factor is smaller than the optimal value, it could make our model do transactions which have little rewards and large risk. If the transaction cost increases and the corresponding weight factor is larger than the optimal value, it could make our model fear to do transactions that have high returns. The relation between commission and portfolio is showed below.

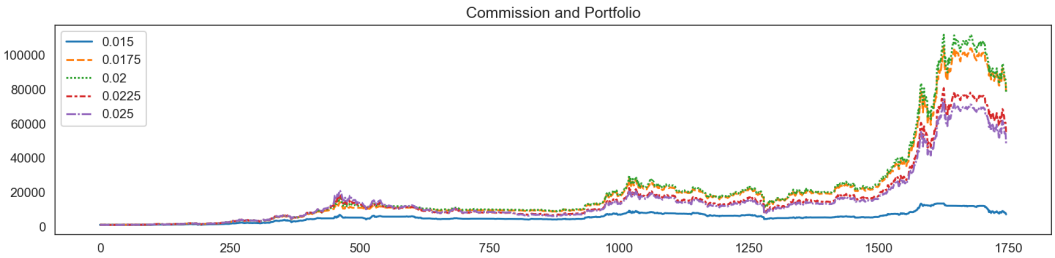


Figure 13: Commission and Portfolio

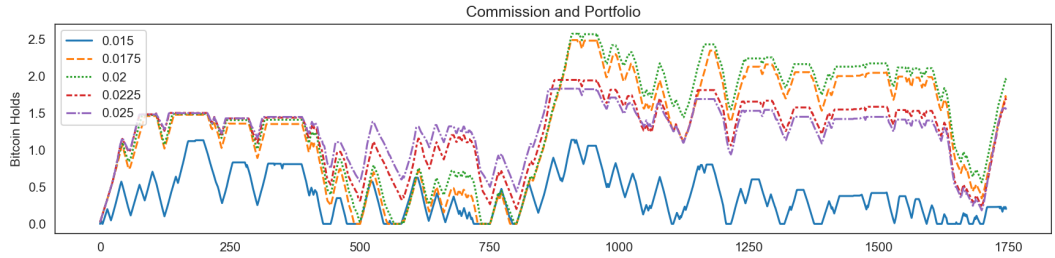


Figure 14: Commission and Portfolio Bitcoin Holds

# 6 Model improvement

The MACD gives the prediction of the market trend, and all following decisions are made based on its signal. However, the MACD is lagged sometimes and may inevitably cause mistakes with a small probability. To enhance the correctness of signals, we modified the decision flow to **Algorithm 3**.

**Algorithm 3** Investment Algorithm (New)

**Require:** $k_{trade}, w_{price}, w_{bull}, w_{risk}, h_{buy}, h_{sell}, k_{trade}$ (can be obtain from the training set)
  **for** each day **do**
    Obtain the trade signal (buy/sell) from MACD model based on the past 60 days of bitcoin and gold price
    Compute TS = Normalize$(w_{price} \times Value_{gain} + w_{bull} \times Value_{bull} - w_{risk} \times Value_{risk})$
    **if** signal = buy **AND** $e^{\text{TS}} > \alpha\ h_{buy}$ **then**
      Buy amount $n_{buy} = k_{trade} \times e^{\text{TS}}$
    **else if** signal = sell **AND** $e^{-\text{TS}} > \alpha\ h_{sell}$ **then**
      Sell amount $n_{sell} = k_{trade} \times e^{-\text{TS}}$
    **end if**
  **end for**

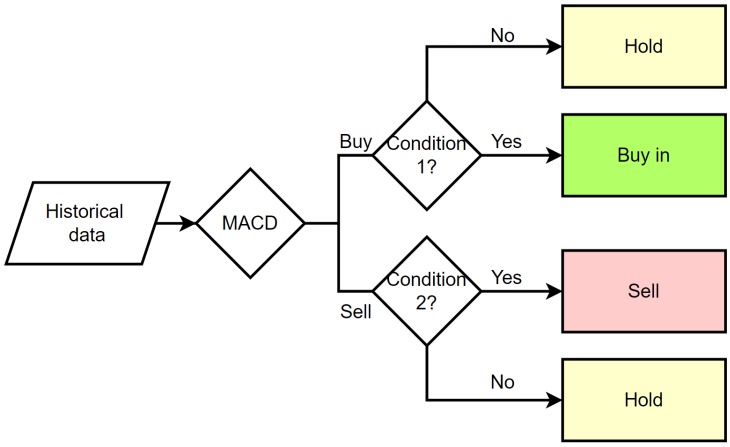


Figure 15: Flow chart of algorithm 3

where condition 1 is $e^{\text{TS}} > \alpha\ h_{buy}$ and condition 2 is $e^{-\text{TS}} > \alpha\ h_{sell}$.

After this revision, the MACD signal is no longer decisive in the decision-making process. Instead, the MACD signal is only a reference provider, and the scoring system becomes the major decider on the actions and trading amounts that takes the MACD's suggestions into consideration.

The robustness of our revised model is improved, and our new portfolio at 2021/9/11 becomes equivalent to **$78417.14**. The trade track is as follows

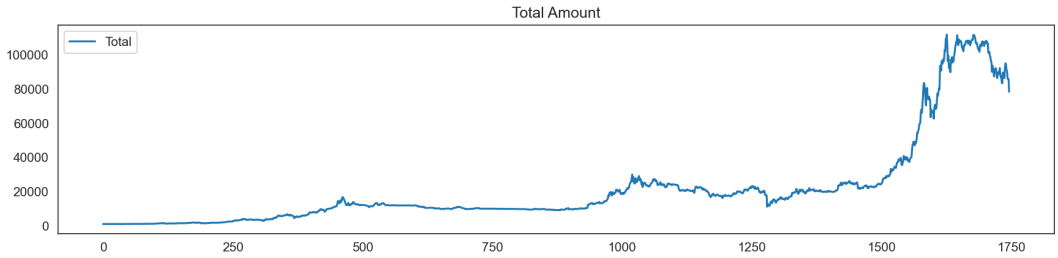


Figure 16: Total account value

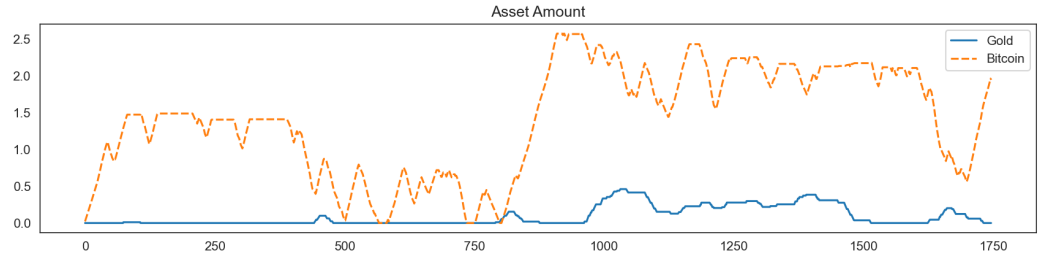


Figure 17: Total asset value

# 7 Strengths and weaknesses of our model

## 7.1 Strengths

- **Integrates multiple approaches for more accurate decisions.**
  We build the model with both data analysis methods and financial knowledge. We try Gray Prediction, ARIMA, LSTM, then find the best model. Also, we combine the theorem in finance, such as BMV and MACD, which help us make the best strategy.

- **Trained on external data and has accurate predictions on the new data.**
  We evaluate the risk, and then make the investment, which makes the strategy more moderate and appropriate. The prediction model fits the given data very well. The MSE of LSTM Prediction is low, showing that LSTM fits well with the real price, and providing an excellent base for the strategy of buy or sell.

- **Large external training dataset avoids overfitting.**
  We used not only the given data of the problem, but also the data set retrieved from the internet that contains the price information from 2011 to 2016. Therefore, our model makes a better strategy at the beginning of the time.

- **Reduces the risk under significance drop of price**
  At the end of 2021, a huge oscillation occurs in the Bitcoin market. Luckily, as the financial models and forecasting techniques foresaw the dropping, we adjusts the proportion of holdings and reduced the loss.

## 7.2 Weakness

- **Attains the best effect only if the market is stable.**
  The gold or death cross are indicators of trend, which assumes the trend is consistent over time. If the future price comes across a sharp rise or fall suddenly, the response may not be very quick. In fact, the adjustment to economic activities of an efficient market mechanism is usually spontaneous and automatic, so this shortcoming exists in every financial analytical technique.

- **Slightly lagged.**
  The time series and the neural network model inclines on the closer dates in their forecasts. The MACD and Bull-bear market evaluation smooths the data by taking weighted averages. Therefore, the response to market adjustments has a few lags.

- **The strategy is a local maximum.**
  We have proved that if the details of our strategy change a little, then the result went down. However, it may not be the global maximum.

# 8 Question 4: Memo to traders

Dear Sir/Madam,
It's our pleasure to introduce you our trading strategy model for gold and bitcoins. In the past few years, the price of bitcoin has experienced a sharp surge. The gold price also has risen up very much. That means, although the prices of gold and bitcoins are of great volatility, the overall tendency is increasing. However, the incorrect strategy won't make a profit and even cause a great loss. To help you make the greatest profit in the price-changing period, our model carries out intelligent analysis to help you make every correct choice in face of the variation trends of the market. The model works in the following steps:

- 1. Predict the price the next day using the historical data.
- 2. Determine the action of each day–buy, sell or hold–after a comprehensive assessment of predicted return and risk.
- 3. Determine the amount to trade or sell by our scoring system.
- 4. Check the optimality and stability of the strategy.

First of all, we build the prediction models of gold and bitcoins for you. We have tried 3 kinds of models on the data from 2008 to 2016, obtaining the parameters that make the models have the smallest mean squared error (MSE). Each of the models can fit the data very well, and most of their errors are lower than 2% of the real value.

Secondly, we developed a scoring standards on whether it is worthy to hold an asset that combines the recent knowledge in Finance and Machine Learning. We apply MACD to be the first checking layer. BMV, the risk to invest, the prediction profit or loss to be another checking layer. Only when the two checking layers make an agreement, the decision to buy or sell will be executed. The operation is under double standards to confirm security.

Third, we provide a good formula to get the trading amount. We found the optimal parameters of the formula based on the price from 2011 to 2016. Then we apply the model to the price set from 2016/9/11 to 2021/9/10, which performed as a test set. The testing result shows that the strategies that the model provides can make a profit of a total of $78417.14 with only $1000 at the beginning of the test set. The graph below shows the amount of the total assets changing with the date with the beginning $1000 in detail.

Finally, we show you the evidence that our model is the most excellent! We put a random noise into the strategy, which affects each operation, and repeat the experiments 1000 times. The distribution of the profit of the experiments is the figure 19.

The result shows that our model beat 95% of the experimental model. This means that if the model is changed, it has a 95% probability to make less profit. Every model can't be a god or prophet, which means it never catches the strategy that the god makes, which is too risky for us. The role of our model is to keep the highest profit under an acceptable

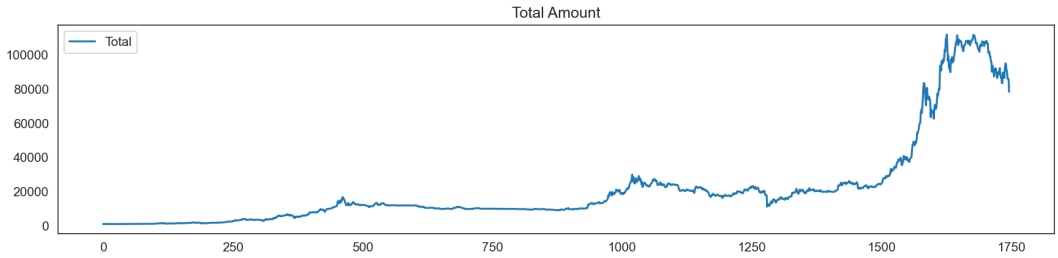


Figure 18: Total asset value

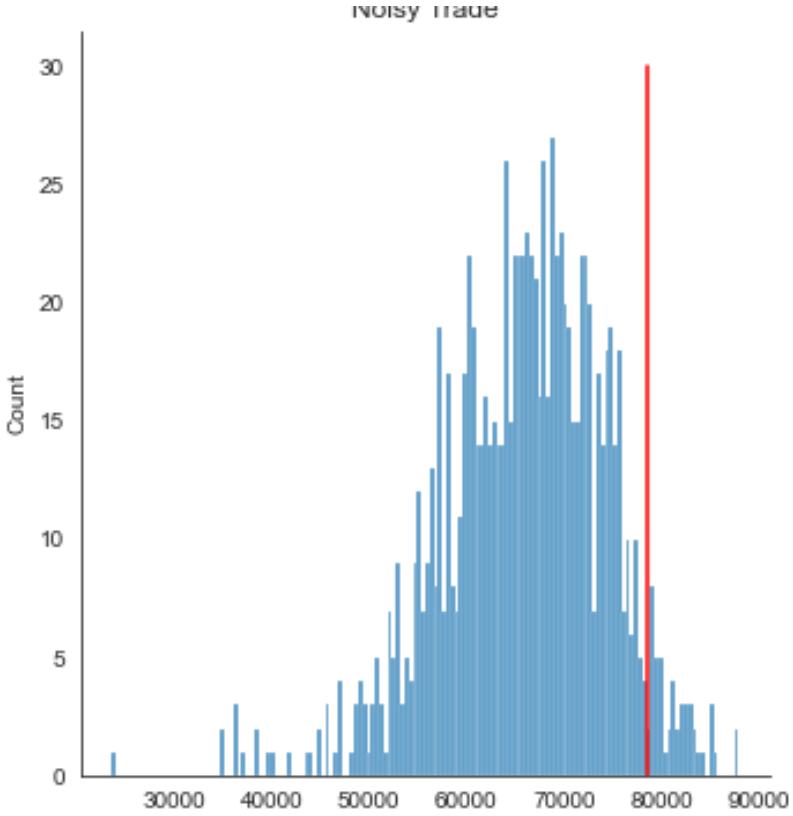


Figure 19: Local maximum

risk. If the trade cost changes a little bit, the total profit won't change too much. So the model is stable, which means it's useful even if changes because of policy or something else about politics.

Last but not the least, if the model is used to predict the future price, you need to input recent data to the model, so that the prediction model can perform well.

Best,

MCM Team 2221952

# References

[1] T. T.-L. Chong, W.-K. Ng, and V. K.-S. Liew, "Revisiting the performance of macd and rsi oscillators," *Journal of risk and financial management*, vol. 7, no. 1, pp. 1–12, 2014.

[2] A. Hameed and Y. Kusnadi, "Momentum strategies: Evidence from pacific basin stock markets," *Journal of Financial Research*, vol. 25, no. 3, pp. 383–397, 2002.

[3] M. K. Kim and J. K. Zumwalt, "An analysis of risk in bull and bear markets," *Journal of Financial and Quantitative analysis*, vol. 14, no. 5, pp. 1015–1025, 1979.

# Appendix

MACD

```
def _ema(arr):
    N = len(arr)
      = 2/(N+1)
    data = np.zeros(len(arr))
    for i in range(len(data)):
        data[i] = arr[i] if i==0 else *arr[i]+(1-)*data[i-1] #
    return data[-1]

def EMA(arr,period):
    data = np.full(arr.shape,np.nan)
    for i in range(period-1,len(arr)):
        data[i] = _ema(arr[i+1-period:i+1])
    return data

def MA(arr, N):
    data = np.zeros(len(arr))
    for i in range(len(arr)):
        data[i] = np.mean(arr[i-N:i])
    return data

class MACD_method:
    def __init__(self, df):
        self. df = df
        self. S = 12
        self.L = 26
        self.position = 0

    def derivative(self, array):
        data = np.zeros(len(array))
        for i in range(len(array)):
            data[i] = array[i] if i==0 else array[i]-array[i-1]
        return data
```

```
   def processing(self):
      self.df['EMA_S'] = EMA(self.df['Price'].values, period=12)
      self.df['EMA_L'] = EMA(self.df['Price'].values, period=26)
      self.df['dif'] = self.df['EMA_S'] - self.df['EMA_L']

      DEA = MA(self.df['dif'].values,12)
      MACD = 2*(self.df['dif'].values - DEA)
      self.df['DEA'] = DEA
      self.df['MACD'] = MACD
      self.df['Signal'] = EMA(MACD, 9)
      # self.df['dK'] = self.derivative(self.df['Price'].values)
      # self.df['dDEA'] = self.derivative(MACD)

   def get_strategy(self):
      self.df['Choice'] = 0
      choice = 'Hold'
      for i in range(3, self.df.shape[0]):
         if self.df['MACD'][i-3] <0 and self.df['MACD'][i-1] > 0:
            choice = 'Buy'
         if self.df['MACD'][i-3] >0 and self.df['MACD'][i-1] < 0:
            choice = 'Sell'
         self.df['Choice'][i] = choice
```

LSTM

```
class PriceDataSet(Dataset):
   def __init__(self, X, y):
      self.X = torch.tensor(X, dtype=torch.float)
      self.y = torch.tensor(y, dtype=torch.float)

   def __getitem__(self, index):
      return self.X[index], self.y[index]

   def __len__(self):
      return len(self.X)

class resLSTM(nn.Module):
   def __init__(self, input_dim, hidden_dim, num_layers, output_dim):
      super(resLSTM, self).__init__()
      self.hidden_dim = hidden_dim
      self.num_layers = num_layers

      self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers,
         batch_first=True)

      self.fc = nn.Linear(hidden_dim, output_dim)
```

```
    def forward(self, x):
        out, _ = self.lstm(x)
        out = self.fc(out[:, -1, :])
        return out + x[:, -1, 0].reshape(-1, 1)
```

ARIMA

```
seq_len = 6
gold_df = pd.read_csv('input\Gold Price.csv')
gold_scaler = MinMaxScaler(feature_range=(-1, 1))
x = gold_scaler.fit_transform(gold_df['USD'].values.reshape(-1, 1))
result_pred = []
for i in tqdm(range(seq_len, len(gold_df))):
    model = ARIMA(x[:i], order=(4,1,4))
    try:
        result = model.fit()
        result_pred.append(result.forecast(step=1)[0])
    except:
        result_pred.append(np.nan)

pred_price = np.array([np.nan] * seq_len + result_pred)
is_nan = np.isnan(pred_price)
mean_squared_error(
gold_scaler.inverse_transform(pred_price[~is_nan].reshape(-1, 1)),
gold_scaler.inverse_transform(x[~is_nan].reshape(-1, 1)), squared=True
)

plt.figure(figsize=(14, 3), dpi=150)
plt.plot(gold_scaler.inverse_transform(pred_price[~is_nan].reshape(-1,
    1)).flatten(),
label = 'Prediction')
plt.plot(gold_scaler.inverse_transform(x[~is_nan].reshape(-1,
    1)).flatten(),
label = 'True Values')
plt.legend()
plt.show()
```

strategy

```
def returnScore(arg, info_df):
    df = info_df[78: -2]
    bit_trade_score = normalize_1D(arg.w_b_price * normalize_1D(df['d
        Bit'].values) + arg.w_b_bull * normalize_1D(df['Bit Bull
        Market'].values) - arg.w_b_risk * normalize_1D(df['Bit
        Risk'].values))
    gold_trade_score = normalize_1D(arg.w_g_price * normalize_1D(df['d
```

```
        Gold'].values) + arg.w_g_bull * normalize_1D(df['Gold Bull
        Market'].values) - arg.w_g_risk * normalize_1D(df['Gold
        Risk'].values))
    return normalize_1D(bit_trade_score), normalize_1D(gold_trade_score)

def buyStrategy(b_score, g_score, b_act, g_act, arg):
    amount = [0, 0]

    if b_act == 1 and np.exp(b_score) > arg.alpha_b * arg.k_a_b:
        amount[0] = arg.n_b_trans * np.exp(b_score)
    elif b_act == -1 and np.exp(-b_score) > arg.alpha_b * arg.k_a_b:
        amount[0] = - arg.n_b_trans * np.exp(-b_score)

    elif np.exp(b_score) > arg.h_b_buy:
        amount[0] = arg.n_b_trans * np.exp(2 * b_score)
    elif np.exp(-b_score) > arg.h_b_sell:
        amount[0] = - arg.n_b_trans * np.exp(2 * b_score)


    if g_act == 1 and np.exp(g_score) > arg.alpha_g * arg.k_a_g:
        amount[1] = arg.n_g_trans * np.exp(g_score)
    elif g_act == -1 and np.exp(-g_score) > arg.alpha_g * arg.k_a_g:
        amount[1] = - arg.n_g_trans * np.exp(-g_score)

    elif np.exp(g_score) > arg.h_g_buy:
        amount[1] = arg.n_g_trans * np.exp(2 * g_score)
    elif np.exp(-g_score) > arg.h_g_sell:
        amount[1] = - arg.n_g_trans * np.exp(2 * g_score)

    return amount

def trade(b_act, g_act, df_info, arg, isPrint = False):
    account = Account(df_info, a_g=arg.alpha_g, a_b=arg.alpha_b)
    b_score, g_score = returnScore(arg, df_info)
    for day in range(len(b_score)):
        account.setDay(day)
        amount = buyStrategy(b_score[day], g_score[day], b_act[day],
            g_act[day], arg)
        if amount[0] > 0:
            account.buyBit(amount[0])
        elif amount[0] < 0:
            account.sellBit(-amount[0])
        if account.isGoldOpen():
            if amount[1] > 0:
                account.buyGold(amount[1])
            elif amount[1] < 0:
                account.sellGold(-amount[1])
```

```
      account.writeState()
      if isPrint:
         print(account.getState())
   return account

def noisyTrade(b_act, g_act, df_info, arg, noisy_std = 0.1, inverse =
   0.05, isPrint = False):
   account = Account(df_info, a_g=arg.alpha_g, a_b=arg.alpha_b)
   b_score, g_score = returnScore(arg, df_info)
   for day in range(len(b_score)):
      account.setDay(day)
      amount = buyStrategy(b_score[day], g_score[day], b_act[day],
         g_act[day], arg)
      if amount[0] > 0:
         if np.random.random() < inverse:
            account.sellBit(np.abs(amount[0] * np.random.normal(1,
               noisy_std)))
         else:
            account.buyBit(np.abs(amount[0] * np.random.normal(1,
               noisy_std)))
      elif amount[0] < 0:
         if np.random.random() < inverse:
            account.buyBit(np.abs(-amount[0] * np.random.normal(1,
               noisy_std)))
         else:
            account.sellBit(np.abs(-amount[0] * np.random.normal(1,
               noisy_std)))
      if account.isGoldOpen():
         if amount[1] > 0:
            if np.random.random() < inverse:
               account.sellGold(np.abs(-amount[1] *
                  np.random.normal(1, noisy_std)))
            else:
               account.buyGold(np.abs(amount[1] * np.random.normal(1,
                  noisy_std)))
         elif amount[1] < 0:
            if np.random.random() < inverse:
               account.buyGold(np.abs(amount[1] * np.random.normal(1,
                  noisy_std)))
            else:
               account.sellGold(np.abs(-amount[1] *
                  np.random.normal(1, noisy_std)))
      account.writeState()
      if isPrint:
         print(account.getState())
   return account
```