Problem Chosen	2022 MCM/ICM Summary Sheet	Team Control Number
C		2222562

Trading Strategies: An Optimal Trading System based on LSTM and Dynamic Programming

Summary

Market Traders always strive to find trading strategies to trade voltage assets to maximize total return. Based on the daily price of Gold and Bitcoin in the recent five years, considering transaction commission, we propose a series of strategies.

First, we treat the floating Bitcoin and Gold prices as two stocks in the data preprocessing part and remove all missing values. Since the future stock prices are correlated with the past prices, we choose time series models, **ARIMA** and **LSTM**.

In the **ARIMA** model, we normalize the data non-linearly by using the log function and then determine the parameters of **ARIMA** by using ACF and PACF graphs. Also, we try to use the dynamic **ARIMA** method. The basic algorithm of this model is to use all previous values as training data and then predict the next day's value. However, after we visualize the prediction results, we find that the predicted values were the latter-day displacement of the original data, which is a case of overfitting. No matter how we adjust the parameters, overfitting could not be avoided, so we give up this method.

For the **LSTM** model, we use linear normalization to get the training data and then get the most suitable combination of hyperparameters by observing the learning curves. Considering the results of the visualized comparison and MSE comparison of two models, we choose to use the **LSTM** model to obtain the predicted assets' daily price.

From the perspective of maximizing investment returns, we also choose the **LSTM** model. Compared to the **ARIMA** model, the **LSTM** model fits a curve with much larger and more frequent fluctuations. In other words, the **ARIMA** curve only shows the long-term trend of Bitcoin and Gold prices, but the **LSTM** curve reflects the daily price changes in more detail. To maximize the investing profits, short-term investments are more advantageous than long-term investments.

After we get the predicted assets daily price, we compare the result from two models with pre-set weights and equal weights check the correlation, covariance matrix, and risks index to prove the validity of setting weight for different voltage assets. Through **Monte Carlo Simulation**, we utilize **Markowitz** model to find the effective weights of asset combinations. Then we apply **Sharpe Ratio** to find out that the most optimal weight of combination for our predicted assets daily price is allocating $188 for Gold and $812 for Bitcoin.

Then we use **Dynamic Programming** strategy to create an Optimal Action Model to find the best dates for trading. Applying the trading timeline created by this model in actual data, we successfully help our assets appreciate from $1,000.00 to $14,140,234.70.

Finally, we test whether our overall model is sensitive to transaction commission and find out the Bitcoins returns decrease 54% when transaction commission changes from 0.01 to 0.03, and Gold returns decrease 21% when transaction commission changes from 0.005 to 0.015. This shows that our overall model is sensitive to transaction commission change.

Our model results propose some confident investing strategies on the portfolio and recommendations for allocating Gold and Bitcoin assets, e.g., the time choice for trading assets, the amount of trading assets, etc. We also write a memorandum for Trade Marketers to summarize our analysis and results, together with our recommendations.

Keywords: LSTM; ARIMA; Dynamic Programming

Contents

1 Introduction

1.1 Problem Background

A profitable volatile assets trading strategy is vital to Market traders. It is always applied to optimize capital allocation to maximize the overall performance, such as expected return. Return maximization is based on the estimates of a stock's potential return and risks. In general, investors make stock investment decisions by predicting the future direction of stocks' ups and downs. In the modern financial market, successful investors are good at using high-quality information to make investment decisions, and they can make quick and effective decisions based on the information they have already had. Thus, the field of stock investment attracts the attention of financial practitioners and ordinary investors, and researchers in academics.

1.2 Restatement of the problem

In this problem, we are given two data sets: the Bitcoin and the daily Gold prices from 9/11/2016 to 9/10/2021 and are asked to develop a model using only the price data up to that day to decide if the trader should buy, hold, or sell their possessions in each day.

The initial possession we will start with on 9/11/2016 is $\$1,000$, and we are trying to maximize the total return in our portfolio, which consists of cash, Gold, and Bitcoin in U.S dollars, on 9/10/2021. We will accomplish the following tasks according to the given data:

- Develop predicted models for the price of Gold and Bitcoins.
- Develop a trading strategy using our predicted model to maximize total return.
- Analyze the sensitivity of our strategy to transaction costs.
- Write a memorandum to the trader by summarizing our trading strategy, model, and result.

1.3 Workflow

Our main goal is to determine the best trading strategy to maximize our profit. To achieve this objective, we first develop several predicted models using Autoregressive integrated moving average (ARIMA) and Long short-term memory (LSTM) to forecast the price of Gold and bitcoin using past data. Then, we compare and analyze these models to find the best price prediction model for further use. Once we successfully predict the price of Gold and Bitcoins, we develop the optimal portfolio model to find the optimal ratio between Gold and Bitcoins in our investment. After that, we combine the optimal weight with dynamic programming to form our optimal strategy model. Last, we obtain the final result through our optimal strategy model.

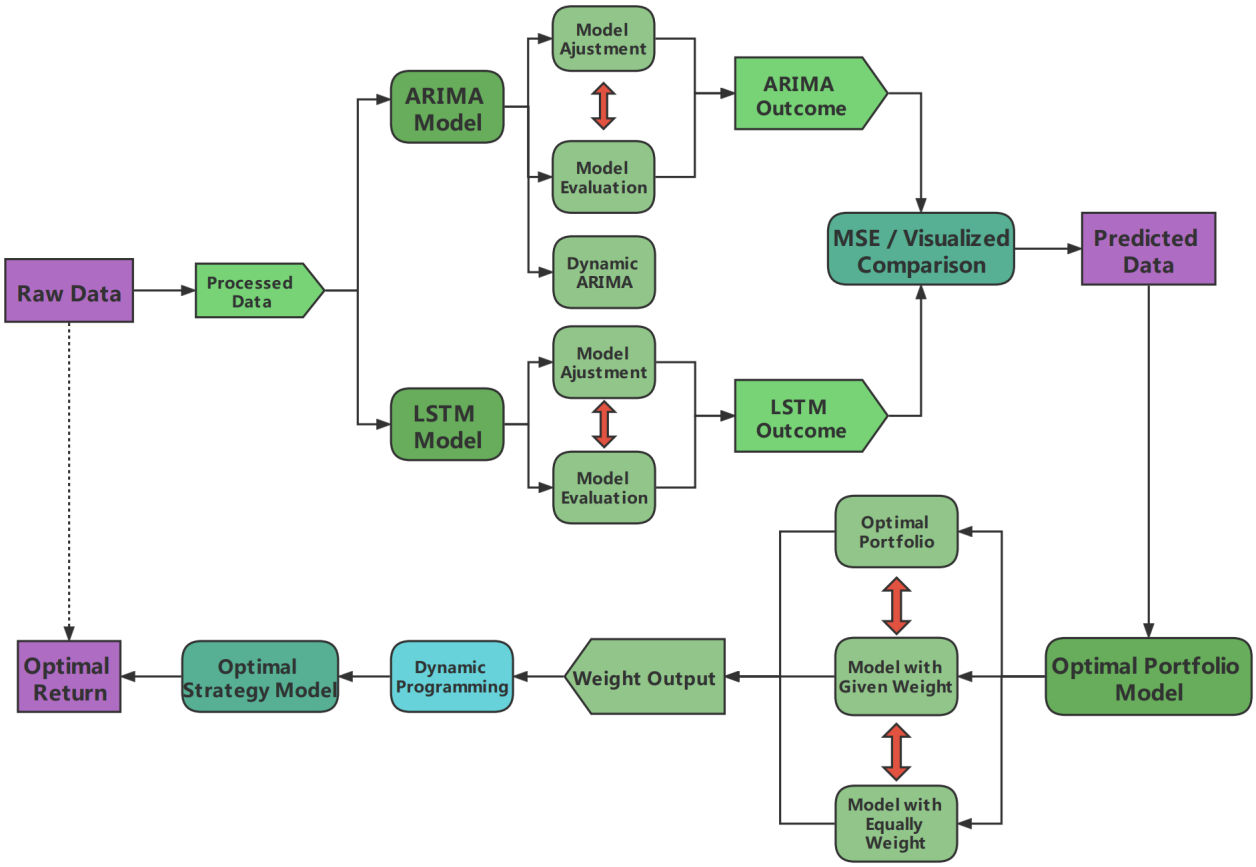

Figure 1: Overall Workflow

1.4 Assumptions

We make the following main assumptions to simplify our model and eliminate the complexity:

- We assume that the U.S. dollar has not experienced inflation in these five years.
- We assume that the volume of trading for Gold and Bitcoins can be fractions.
- We assume that fractions of cents will not be truncated during transactions.
- We assume that the given data is the closing price each day.

1.5 Notations

Table 1: Variable Description

Symbol	Definition
L	Lag operator
ϕ_i	Parameter of autoagressive part of ARIMA
θ_i	Parameters of moving average part of ARIMA
ϵ_t	Error terms of ARIMA
C_t	Cell state
U_t	Update filter
O_t	Cell state that is going to output
h_t	Hidden state to be passed on next cell
σ_p	Standard deviation of portfolio
$\sum$	Covariance matrix of returns
ω	Portfolio Weights (ω_T is transpose of portfolio weights)
R_s	Sharpe Ratio
R_p	Expected rate of return
R_f	Interest Rate with no risk
σ_r	Standard deviation of excess returns
M_t	Cash Amount at time t
S_t	Stock Amount at time t
X_t	the tth data using the ARIMA model
e_t	the tth error term using the ARIMA model
Y_i	the ith real value
$\hat{Y}_i$	the ith predicted value
n	data size

2 Data Prediction

2.1 Data Preprocessing

First, we preprocess both Bitcoin and Gold data sets. When looking at the Gold data, we find that there are missing values (NA value) in the price column of the Gold data. Considering that the Gold data has only two dimensions (date and price) and the original Gold data has only ten missing values for 1265 rows, we simply ignore the missing values. In other words, we only calculate 1255 Gold data.

Based on our initial data analysis, we can consider the daily floating values of Bitcoin and Gold as two stocks. As predictions in stock trading require the consideration of previous data, we choose to use time series models as our primary forecasting models. We fit, analyze, and compare two models: the autoregressive integrated moving average (ARIMA) and the long short-term memory (LSTM) models.

To better compare and evaluate the strengths and weaknesses of the models, we chose to use the Bitcoin data, which is more variable than the Gold data, as our training and testing data. We used the first 80% of this data (index numbers from 1 to 1460) as training data and the last 20% (index numbers

from 1461 to 1826) as testing data to visualize the comparison between predicted and true values. Also, in order to come up with a more accurate model, we choose not to do any trading operations in the first five days but just record the data. In other words, our predicted values start from the sixth day. Figure 1 shows the overall process of data prediction.

2.2 AutoRegressive Integrated Moving Average (ARIMA)

2.2.1 Introduction to the Model

ARIMA is a combination of two models, the Auto-Regressive model (AR) and the Moving Average model (MA). AR demonstrates the consideration of past values in the regression equation of the series Y, while MA represents the model's error as a combination of previous error terms. The auto-regressive parameter p specifies the number of lags, the moving average nature of the model is defined as parameter q, and the differencing variable d is used to remove the trend and convert a non-stationary time series to a stationary one. Thus, ARIMA$[p, d, q]$ is defined as

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d X_t = \delta + (1 + \sum_{i=1}^{q} \theta_i L^i)\epsilon_t,$$

where L is the lag operator, ϕ_i are the parameters of the autoregressive part of the model, θ_i are the parameters of the moving average part, and ϵ_t are error terms.

2.2.2 Adjustment of the Model

First, we choose to plot a trend graph of the current data (Figure 2). From the left graph in Figure 2, the series is not yet smooth, so we choose to normalize the original data with log function and do differencing to make the trend smoother, shown on the right graph in Figure 2. We also conduct the Dickey-Fuller test to check the stationary, and the test returns a p-value of 0.01, which means we reject the null hypothesis and accept the alternative: the data is stationary[5].

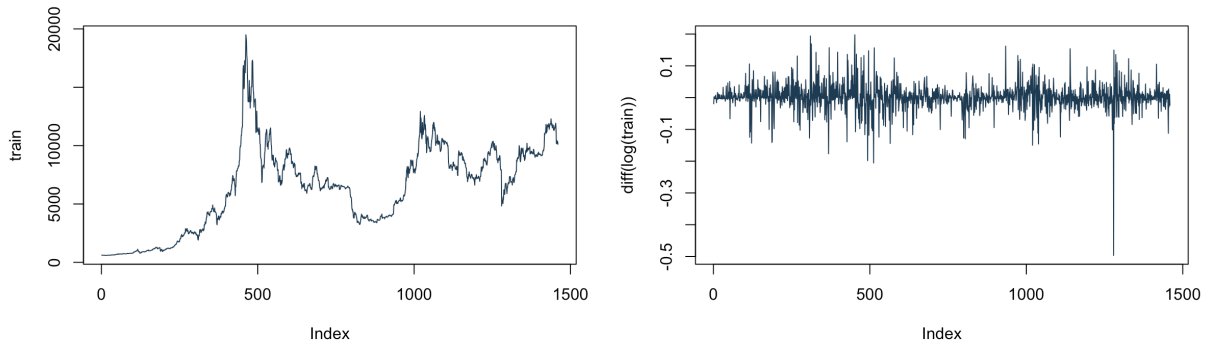

Figure 2: Plot of the original Training Data and the Training Data after normalizing and differencing

To determine the order of our parameters, we look at the difference in lags in both the ACF and PACF graphs (Figure 3 and Figure 4 respectively). In the ACF graph (Figure 3), the curve drops significantly after the first lag, so we should model with one moving average component ($q = 1$). For the PACF graph (Figure 4), it has a significant cut-off after 1st lag, which indicates the time lag tends to be one ($p = 1$). Thus, the optimal model may be ARIMA[1, 0, 1][7].

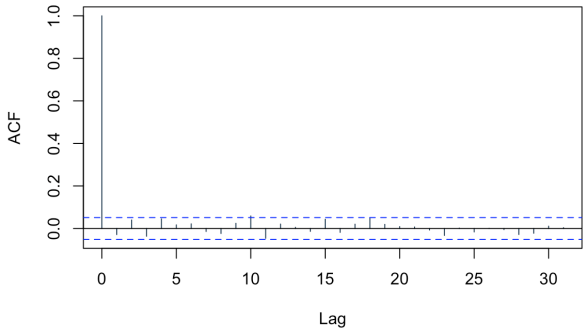

Figure 3: ACF of the Training Data

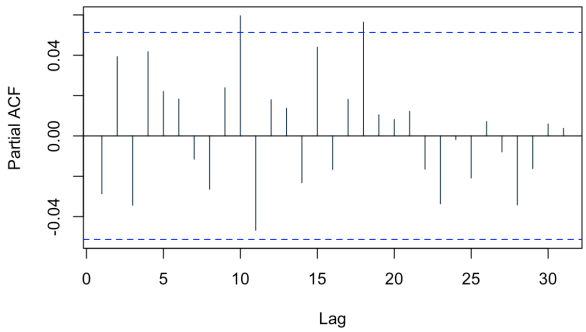

Figure 4: PACF of the Training Data

We use the auto.arima() function in R to get the best ARIMA model of the training data. It returns the same model as we predict, shown on the left table of Table 2). We can also see that the AIC and BIC values of the model are quite low, which indicates that the model performs well.

```
Series: diff(log(train))
ARIMA(1,0,1) with non-zero mean

Coefficients:
          ar1     ma1    mean
      -0.8029  0.7658  0.0019
s.e.   0.1408  0.1511  0.0011

sigma^2 = 0.001764:  log likelihood = 2556.26
AIC=-5104.53   AICc=-5104.5   BIC=-5083.39
```

```
        Ljung-Box test

data:  Residuals from ARIMA(1,0,1) with non-zero mean
Q* = 11.017, df = 7, p-value = 0.1379

Model df: 3.   Total lags used: 10
```

Table 2: Outcome of the ARIMA Model and Result of Ljung-Box Test

To test the ARIMA model, the first thing we do is determine the model's residuals. From the graph in the lower right corner of Figure 5, we find that the residual distribution satisfies normality. That is, the residuals of the model are concentrated at a value near 0, and this value obeys a normal distribution with the mean zero, which means the value is white noise. Next, we consider the Ljung-Box test for the model, shown on the right table of Table 2. Since the null hypothesis is that the original series is a white noise series, we consider that the correlation coefficient is not significantly different from zero when we obtain a p-value greater than the significance level of 0.05 or 0.1. In this model, the p-value we obtained for the Ljung-Box test is 0.1379 (greater than 0.1), so we consider the original series a white noise series. In short, the model passes the test.

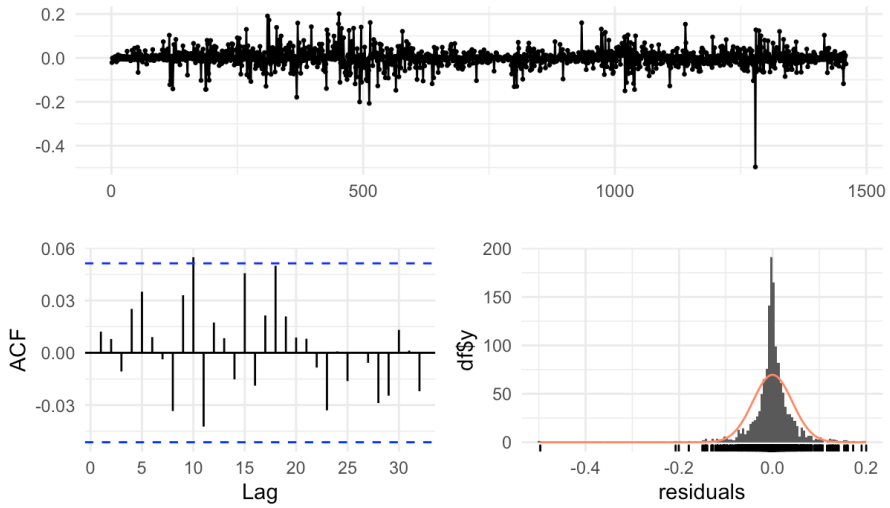

Figure 5: Residuals of the ARIMA model

In order to obtain a more accurate model, we also try to dynamic the ARIMA model. The logic is that we put all the data from previous days into the ARIMA model as training data and come up with a prediction for the next day. In this case, the ARIMA model is dynamic because the training data is constantly being updated day by day. For example, on the closing of the fifth day, we use the ARIMA model to predict the data of the sixth day using the data of the previous five days. Then, this is repeated several times until the whole data (1826 days Bitcoin price) except the first five data is predicted. We record all the predicted data and then plot a graph to compare the predicted data to the original data, shown on the left graph of Figure 6.

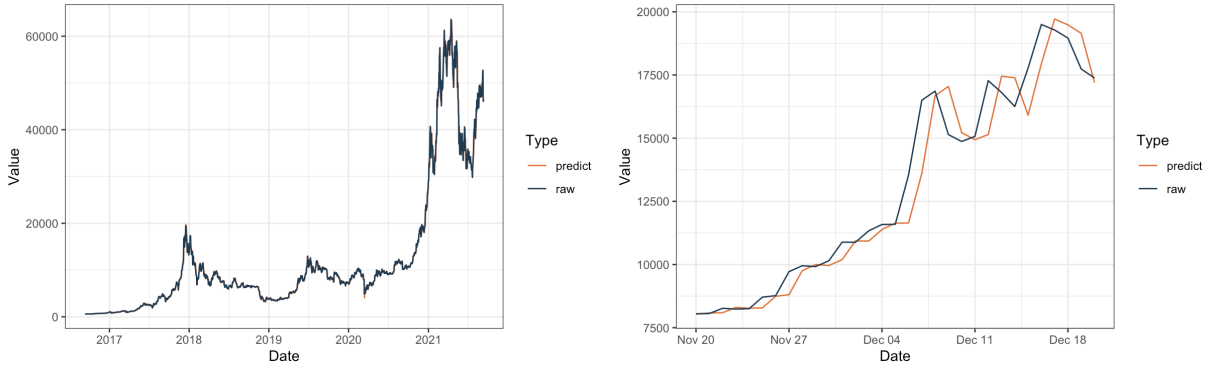

Figure 6: Plot of Predictions from the Dynamic ARIMA model

We hardly see the presence of the predicted line in Figure 6. That is to say, the predicted and raw values almost overlap. For the right graph of Figure 6, we zoom a part of the prediction graph, and it is clear that the predicted data is almost the same as the original data shifted one day to the right. In other words, our new predicted value is very close to the last value of the training data. The reason for this situation is overfitting. Dynamic ARIMA uses a large amount of data to predict one future day's value, resulting in a low biased model but has a high variance (the model is too detailed and has low generality). In fact, we even tried to increase the number of predicted days from 1 to 20, but the results

still suffer from overfitting.

2.2.3 Model Outcome

Getting the coefficients from the left table of Table 2, we can get the final mathematical equation for the model from ARIMA (for the data after doing the log and differencing operations):

$$(1 + 0.8029B)(X_t - 0.0019) = (1 + 0.7658B)e_t,$$

So

$$X_t = 3.425 \times 10^{-3} - 0.8029X_{t-1} + e_t + 0.7685e_{t-1},$$

where X_t represents the tth data, and e_t represents the tth error term.

We visualize the final outcome we get from that model (Figure 7).

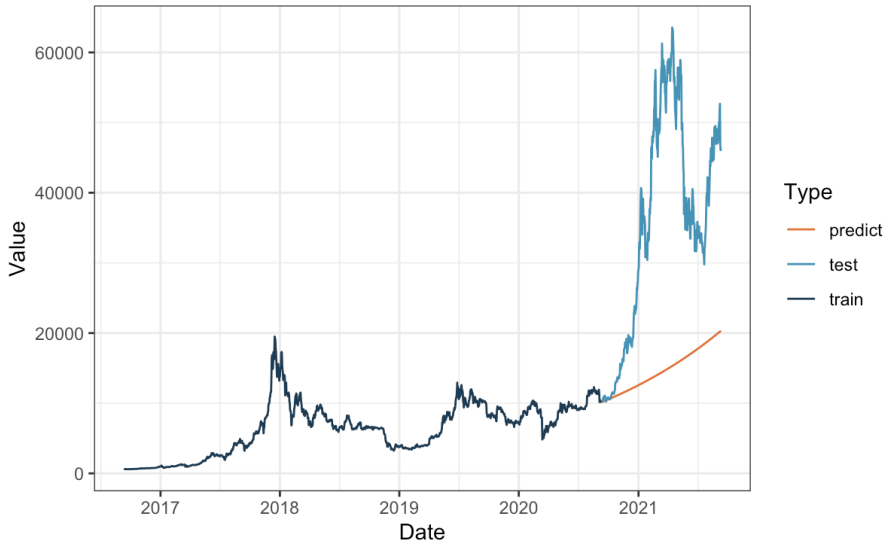

Figure 7: Plot of Predictions using the ARIMA Model

2.3 Long Short-Term Memory (LSTM)

2.3.1 Introduction to the Model

Long Short-Term Memory (LSTM) is a special type of Recurrent Neural Network (RNN) capable of learning long-term dependencies. In RNN, because there is a recursive effect, the state of the hidden layer at the last moment is involved in the computation process at the present moment. That is to say, the selection and decision are made regarding the previous state. LSTM inherits this advantage.

LSTM typically consists of several memory blocks, known as cells, connected through different layers. Gates control the information stored in the cell and hidden states through activation functions like sigmoid and tanh. In general, the gates take the hidden states from the previous step h_{t-1} and the current input x_t and multiply them pointwise by weight matrices W, and a bias b is added to the model. There are three main gates (using tanh as an example):

- Forget gate. It determines what information to delete:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

- Input gate. It determines which values in the input are used to update the memory state:

$$\hat{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

$$U_t = \sigma(W_u[h_{t-1}, x_t] + b_u)$$

$$C_t = f_t \cdot C_{t-1} + U_t \cdot \hat{C}_t$$

- Output gate. It determines the value of the output based on the input and memory state:

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = O_t \cdot \tanh(C_t)$$

(In the formulas above, C_t represents the cell state, U_t refers to the updated filter, O_t is the cell state that is going to output, and h_t stands for the hidden state to be passed on to the next cell)

2.3.2 Adjustment of the Model

Before feeding the data into the model as training data, we need to normalize the original data. There are two reasons for doing the normalization. First, normalization can improve the speed of the gradient descent method to solve the optimal solution. Since LSTM is developed based on RNN, the essence of LSTM is to minimize the loss by gradient descent method to obtain the optimal solution. Applying normalization to the data in the gradient descent method can help the model reach the convergence state faster. Second, normalization has the potential to improve accuracy. According to the characteristics of the original data (the values are relatively concentrated), we do not consider standard deviation normalization and nonlinear normalization. Instead, we choose linear normalization. The formula is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}.$$

Before training a model for machine learning, we need to choose the right hyperparameters. In LSTM models, a few essential hyperparameters are shown below:

- Epoch: this is the total number of model forward or backward propagation iterations.
- Number of hidden layers: the number of hidden layers of the neural network. Although our input data is of low dimensionality (Value is the only dimension), we still choose the number to be 10 to get better predictions.
- Batch Size: this is the number of training samples during one forward or backward propagation before the weights are updated. The batch size must be the common factor of the training and test sets. Since the length of our training and test sets is 1455, we choose a factor of 97.

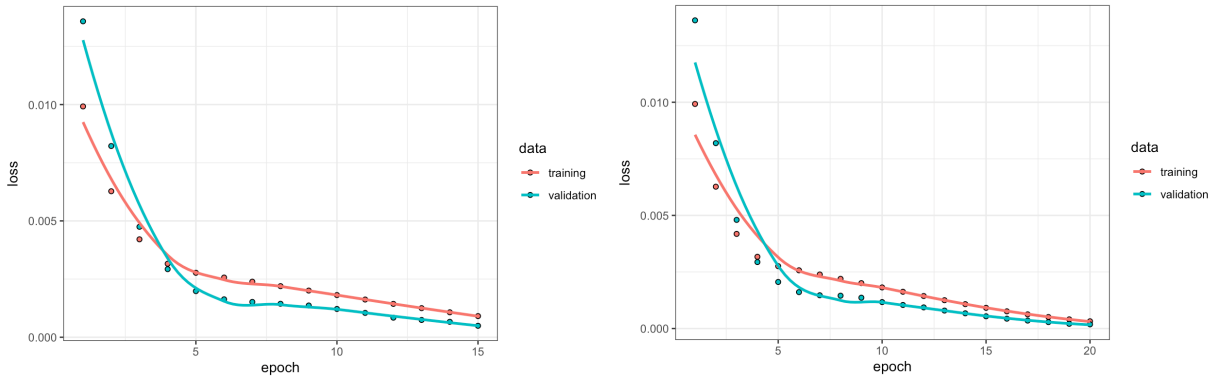

Figure 8: Learning Curves of the LSTM Model when Epoch = 15 and Epoch = 20

- Time step: the lag length between the training and test sets. In this case, we choose 5, which means that the overall data is considered with a lag of 5 days.

Figure 8 is when we take Epoch equal to 15 and when Epoch is equal to 20. We can see that the training loss and validation loss are almost equal. When Epoch exceeds 20, the model will be overfitting, so we finally determine Epoch to be 20.

In the time series, cross-validation is not easy to do. We cannot choose random samples and assign them to either the test set or the train set. The reason is that we may choose a value from the future to test a value from the past. That situation makes no sense. In simple words, we want to avoid future-looking when we train our model.

2.3.3 Model Outcome

The visualization of the final LSTM model is shown on the left graph of Figure 9. In the plot, "raw" represents the original data, "train" means the fitted values of original training data, and "test" indicates the prediction values of the model. For the LSTM model, we can hardly see any original data, which means the overall performance of the model is great. The right graph of Figure 9 shows the partial enlargement. Although there are still some subtle differences between the original and predicted data from day to day, the overall trend is successfully simulated.

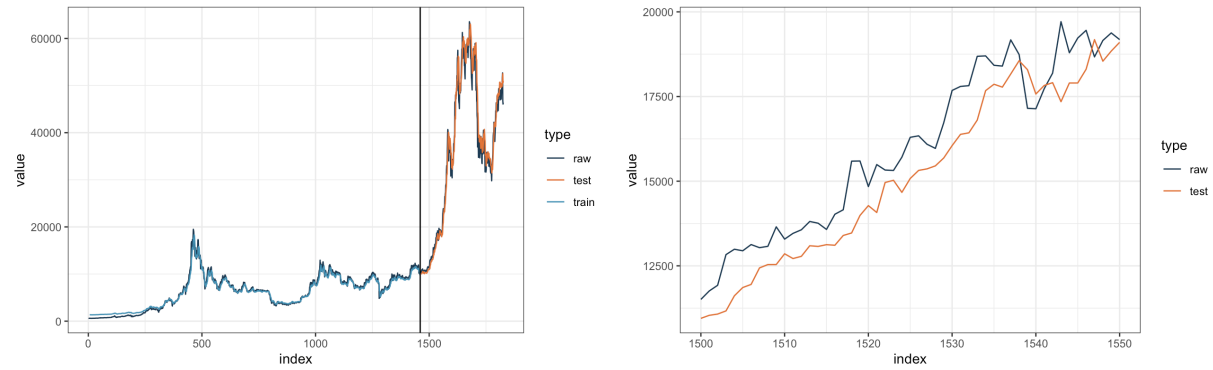

Figure 9: Plot of Predictions using the LSTM model

3 Model Assessment

3.1 Visualized comparison

In order to horizontally compare the predictions from different models for the same set of data, we first plot the original data, the ARIMA prediction data, and the LSTM prediction data on one graph (Figure 10). The orange line represents the original Bitcoin dataset, the blue line is the predictions of ARIMA, and the dark blue line is the predictions of LSTM. From the left graph of Figure 10 it is easy to see that LSTM has better prediction results than ARIMA. ARIMA is more like a monotonically increasing polynomial function than LSTM. The right graph of Figure 10 is an enlarged graph of the right graph, starting from the first date of the test data (the last 20% of the original Bitcoin data).

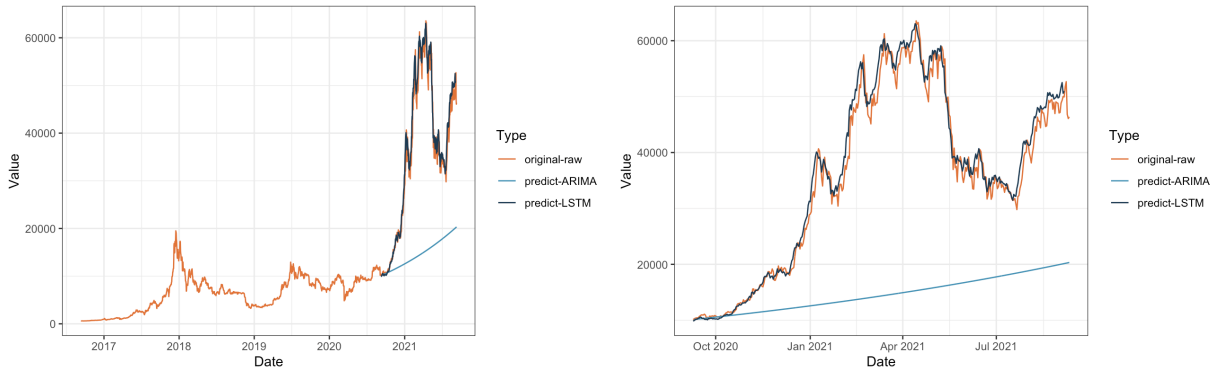

Figure 10: Visualization of Comparison between Predictions using the ARIMA Model and LSTM Model

From an algorithmic point of view, we would also choose to use the LSTM model because the fitted line of the ARIMA model is too smooth compared to that of the LSTM model. However, in order to achieve maximum returns, we focus more on short-term investments (which will be discussed in detail later). An overly smooth curve does not meet our needs because it can only reveal trends over a large time horizon (say six months to a year) and cannot show specific daily changes.

3.2 Evaluation with MSE

To compare the model more logically and accurately, we use Mean Square Error (MSE) to evaluate the model, an essential metric of the model accuracy because it calculates the mean of the sum of squared difference between all predicted values and true values. The formula for MSE is

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2,$$

where n is the data size, Y_i is the real value, and $\hat{Y}_i$ is the predicted value.

The MSE of the ARIMA model is 635167400, while the MSE of the LSTM model is 6757875, shown in Table 3. Therefore, we choose LSTM as our final model to accomplish the data prediction.

We then apply LSTM to the Gold data, fit the model (Table 4), and visualize the predictions (Figure 11). The MSE for this model is quite low, which indicates that the accuracy of the predicted Gold prices is high.

```
46/46 - 0s - loss: 9.2843e-05 - 450ms/epoch - 10ms/step
12/12 - 0s - loss: 0.0017 - 71ms/epoch - 6ms/step
[1] "Train Score: 368031.6581 MSE (606.6561 RMSE)"
[1] "Test Score: 6757874.7840 MSE (2599.5913 RMSE)"
```

Table 3: MSE and RMSE of the LSTM model for the Bitcoin Data

```
32/32 - 0s - loss: 5.0344e-04 - 464ms/epoch - 15ms/step
8/8 - 0s - loss: 8.5752e-04 - 42ms/epoch - 5ms/step
[1] "Train Score: 446.2142 MSE (21.1238 RMSE)"
[1] "Test Score: 760.0398 MSE (27.5688 RMSE)"
```

Table 4: MSE and RMSE of the LSTM model for the Gold Data

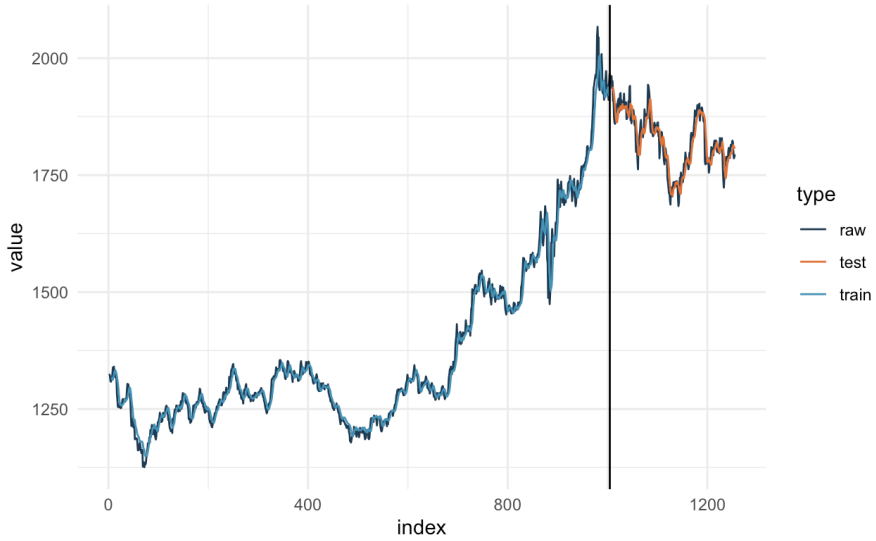

Figure 11: Plot of Predictions of the Gold Data using the LSTM model

4 Optimal Portfolio Model

Financial markets are fraught with uncertainty. Since we have to decide the time to trade stocks based on the results of our forecasting models, thus we should consider increasing return rates while reducing the risk of our investments. The core task in our investment optimization is to find out how an investor can allocate the assets to achieve maximizing (cumulative) returns for a given risk. In this section and next section, we will introduce our optimal portfolio model for predicting optimal weights and optimal action model for predicting time to trade.

We first get the predicted price of each stock to get the daily return in these five years. When deciding how to allocate the funds and trade stocks, we need to set the appropriate weights for each stock trading. In other words, when trading stocks, we should allocate money by using 50% for Gold and 50% for Bitcoins or using 30% for Gold and 70% for Bitcoins. Regarding this problem, we have the following three weighting schemes.

4.1 Portfolio with Given weights

Before trading, we have been given some preset weights in these weighting schemes, for example, 40% for Gold and 60% for Bitcoins. Furthermore, we get the daily return of stocks with such a weighting method. The result for the stock combination model is shown in Table 5.

dates	gold	bits	Portfolio
1	-0.001567	-0.001028	-0.001244
2	-0.001466	-0.000690	-0.001000
3	0.000728	0.000986	0.000883
4	0.004708	-0.000372	0.001660
5	0.004374	-0.000910	0.001203

Table 5: Daily Return of Predicted Data

Also, we create Daily Return Rate (the left graph in Figure12) and Cumulative Return Rate (the right graph in Figure12) to figure out the returns of the portfolio with the given weight (Gold : Bitcoins = 0.4 : 0.6).

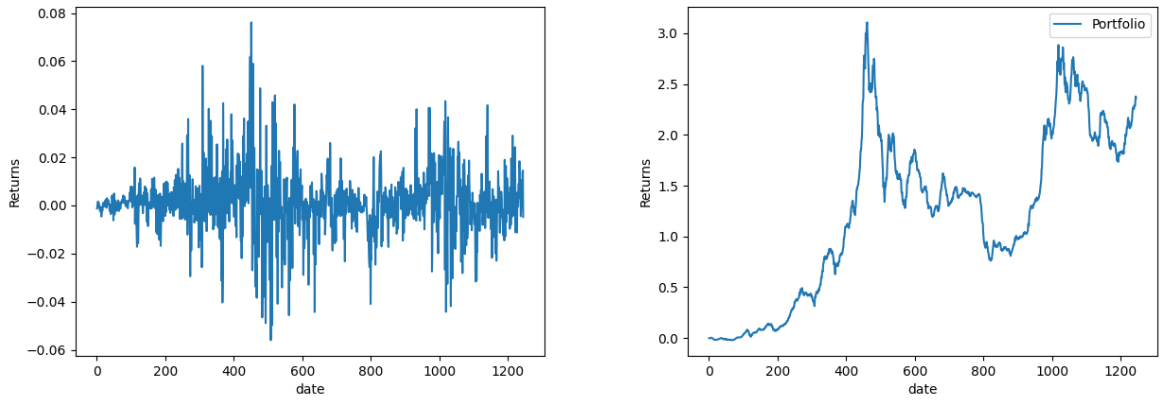

Figure 12: Daily Return Rate(left) and Cumulative Return Rate(right) with given weight(Gold : Bitcoins = 0.4 : 0.6)

4.2 Equally-weighted Portfolio

In these weighting schemes, we equally allocation money on both stocks. That is to say, allocating 50% money on Gold stock and 50% money on Bitcoin stock. Although this method rarely becomes the optimal portfolio, we can use this as a reference benchmark for other portfolios. In Figure 14, we plot the cumulative return rate change corresponding to time in the same graph. The blue line denotes the portfolio in the first method, given the weighting scheme(Gold : Bitcoins = 0.4 : 0.6), and the tangerine red line shows the equally-weighted scheme. The first weighting scheme is better than the equally-weighted portfolio in this situation.

Moreover, we need to check the correlation of the two assets, Gold stock and Bitcoin stock, since it is essential for deciding the portfolio in the financial market. For example, investors can achieve diversification benefits by adding low or negatively correlated mutual funds to an existing portfolio. In other words, if the correlation between two assets is negatively correlated, investors can use them to hedge their portfolios and reduce market risks due to volatility or shape price swings. The following Table 6 and Figure 13 shows the correlation between Gold and Bitcoins.

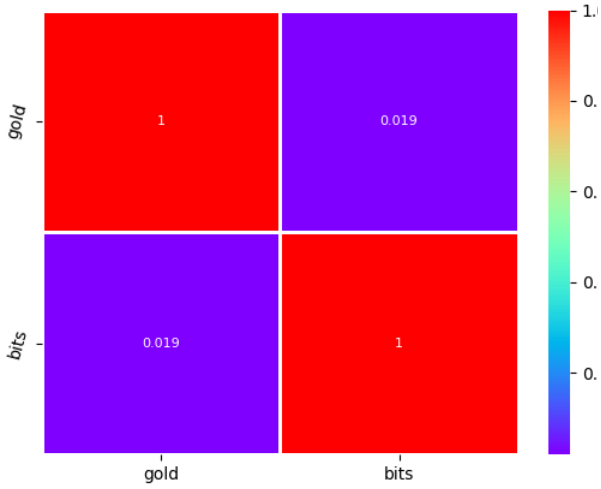

Figure 13: Hot Image

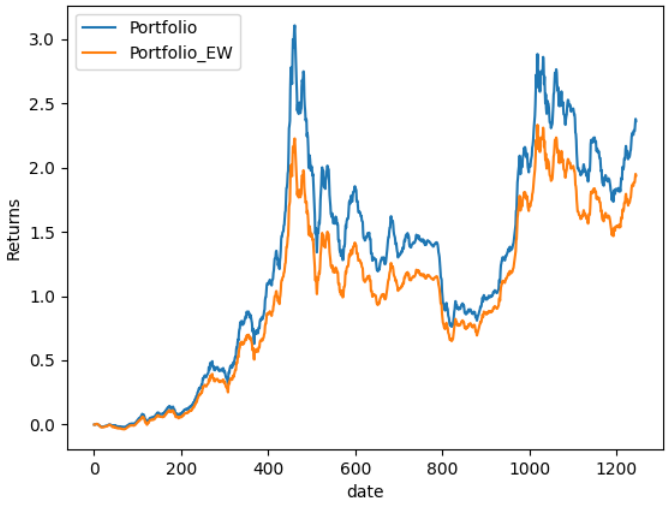

Figure 14: Equal Weight vs. Set Weight

	gold	bits
golds	1.000000	0.019372
bits	0.019372	1.000000

Table 6: Correlation Table

Each element in the matrix is the correlation index of that stock, ranging from -1 to 1, a positive index denotes a positive relationship, and a negative index denotes a negative relationship. The diagonal elements in the matrix are always 1 since the stock and itself are always a totally positive relationship. The correlation between gold and stock is about 0.02, meaning these two assets have almost no relationship with each other. Thus, it proves our validity in operating two assets separately. Besides correlation which only shows the linear relationship between assets, not telling the volatility of assets, we need to introduce a covariance matrix to tell us that information.

	gold	bits
golds	0.003876	0.000409
bits	0.000409	0.114935

Table 7: Covariance Matrix

From Table 7, we find out Bitcoin has apparent volatility since the covariance for Bitcoin is 0.115, which is a great number for covariance. After exploring the above two coefficients, we should find out the standard deviation to express the risks of the investment combination after knowing the weights and covariance matrix. The following equation is the formula for the method, where σ_p is the standard deviation of a portfolio, $\sum$ is the covariance matrix of returns, and ω is portfolio weights (ω_T is the transpose of portfolio weights).

$$\sigma_p = \sqrt{\omega_T \cdot \Sigma \cdot \omega} = 0.2054$$

4.3 Exploring the optimal portfolio of stocks

From the above two weighting schemes, we can see pre-set weights proves better than equally-weight scheme. However, finding the best weights is still a problem to be solved. Considering how to balance the returns and risks when choosing the investment combination, we can introduce the Markowitz model to analyze the data and determine the best weights for predicted data.

4.3.1 Monte Carlo Simulation

Markowitz model [6] works well here because the background of the problem perfectly fits the prerequisite of the model. The investor considers each investment choice based on the probability distribution of the returns of the securities over a given holding time. The investor estimates the risk of a portfolio of securities based on the expected rate of return. The investor's decision is simply a sentence about the risk and return of the security. At a certain level of risk, the investor expects to benefit the most. Moreover, we decide to introduce Monte Carlo Simulation to create random weights to compare the outcome from different weights.[4]

When we use Monte Carlo Simulation to analyze, we randomly create a set of weights to calculate the returns and standard error of the return and repeat this progress many times(10000 times in our model).[3] Through this method, we take in each return and standard error into the model as a point to construct the scatter plot, Figure 15.

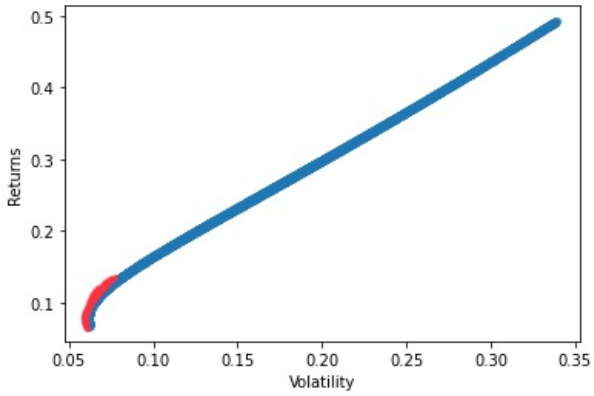

Figure 15: Simulation Result

The nature of investment is to choose the balance between risks and returns, and Figure 15 depicts the two elements. Each point in the graph shows a portfolio combination, the x-axis shows the standard deviation of risks, and the y-axis is return rates.[2] Markowitz investing combination rule considers the wise investor is always maximizing the returns given fixed risk, or minimizing the risk given fixed return. It is shown in the graph as red edge, whose points on edge are the most effective investment combinations. Now we find out a series of most-effective investment weight combinations. However, we need to choose a strategy to find a final weight for Gold and Bitcoins. Here we introduce and compare two strategies: Investment risk minimization portfolio (minimize the risks) and Optimal portfolio (maximize the returns with uncertain risks).

4.3.2 Investment risk Minimization Portfolio

One strategy is to find the highest return in the lowest risk situation, which is called Global minimum volatility(GMV) portfolio[1]. We successfully find this combination and draw it in the Figure 16.

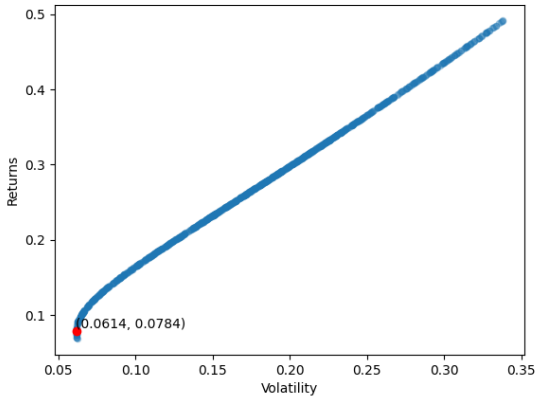

Figure 16: GMV Portfolio

4.3.3 Optimal portfolio of Investments

Since we use the optimal portfolio of investments here, we have to admit that certain risks will show up, while An wise investor can always burden certain risks to strive for a higher return. Thus, we will introduce Sharpe Ratio[1] here to help us balance return and risks for each investment combination.

$$R_s = \frac{R_p - R_f}{\sigma_r}$$

where R_s is the Sharpe Ratio, R_p is expected rate of return, R_f is interest Rate with no risk, σ_r is the standard deviation of excess returns. The numerator calculates the spread, the excess return of an investment compared to a benchmark representing the entire investment portfolio. The denominator standard deviation represents the return volatility and responds to the risk, as higher volatility predicts higher risk. We can simply divide the mean of the excess return by the standard deviation, which is the Sharpe ratio measuring return and risks, and multiply it by $\sqrt{252}$ (there are 252 trading days in a year) to get the annualized Sharpe Ratio.

Then, we add Sharpe Ratio as the third variable into the return-risk scatter plot, Figure 17, and we use color to show Sharpe Ratio here. We find that the upper edge has higher Sharpe Ratio here. Thus, we should figure out the combination with greatest Sharpe Ratio in the scatter plot and find the weight of that combination, as shown in Figure 18.

[1]The ratio is the average return earned in excess of the risk-free rate per unit of volatility or total risk.

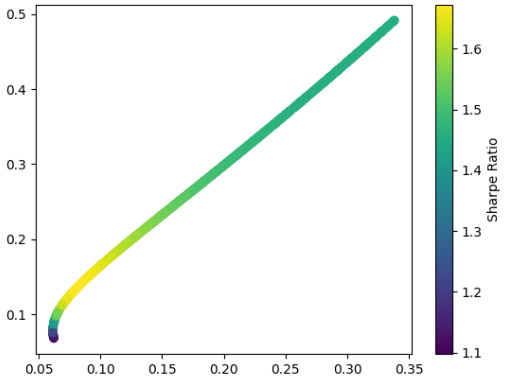

Figure 17: Sharpe Ratio Plot

4.4 Model Outcome

In this section, we introduce and compare three weighting schemes: portfolio with given weights, portfolio with equal weights, and optimal portfolio weights. We utilize Monte Carlo Simulation, Markowitz model, and Sharpe Ratio to determine the best weights for predicted data. Through these steps, we get the optimal weights for Gold and bits is $0.188 : 0.812$. That is to say; we decide to allocate \$188 for Gold investment and \$812 for Bitcoins investment.

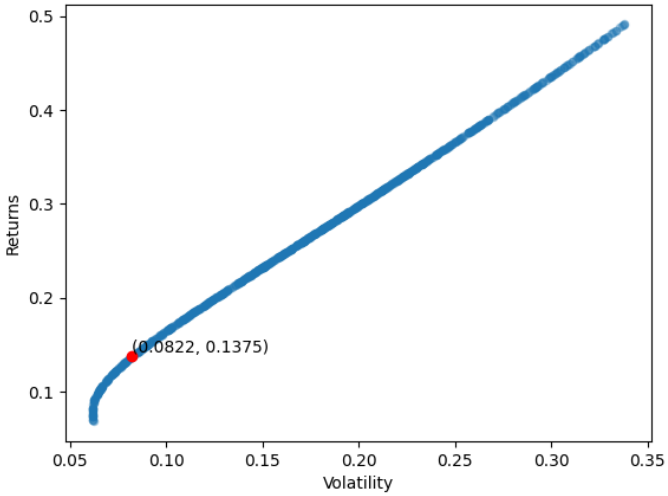

Figure 18: Optimal Combination

	gold	bits
weight	0.188	0.812

Table 8: Optimal Weight Table

5 Optimal Strategy Model

5.1 Introduction to Dynamic Programming Model

From the optimal portfolio model result, we decide to use $188 to make investments in Gold and $812 to make investments in Bitcoins. In this way, we can avoid the difference in trading days since only Bitcoins can be traded at weekends. When we decide to separately invest the stocks, in order to get greater return, if we find a model that assists us maximize either investment, the final return rate will definitely the highest return rate.

We decide to choose the dynamic programming model to find out the best time to trade since this problem meets the optimality principle[2]with the overlap of subproblems[3] and no posteriority[4]. In this model, in order to maximize the return rate for each day, we need to compare and choose the the higher return rate for two choices: buy the assets or sell the assets. From this guideline, we can gradually find the optimal operation from start day to the last day.

In this model, M is as the cash, α is as transfer fee, S is as the stock. We first set the start of cash (M_0) in hand as 1, and the start of stock as the maximum stock (S_0) can be bought using M_0.

$$M_0 = 1 \qquad S_0 = \frac{1 \cdot (1-\alpha)}{P_0}$$

The state transfer function will be as follow:

$$M_t = \max(M_{t-1}, S_{t-1} \cdot P_t(1-\alpha))$$

$$S_t = \max(S_{t-1}, \frac{M_{t-1} \cdot (1-\alpha)}{P_t})$$

5.2 Model Outcome

We predict the best time to trade from our Dynamic Programming (DP) Model and the predicted stock price from the LSTM model. To better show the operation timeline, we mark the time to sell stock as a green circle and the time to buy stock as an orange triangle in the following Figure 19 and Figure 20. After applying the time information given by our model, we make our investment change from 1,000.00 dollars to 14,140,234.70 dollars, which is a big success.

[2]Regardless of the past states and decisions, the remaining decisions must constitute the optimal strategy for the state formed by the previous decisions.

[3]Thesub-strategiess of an optimal strategy are always optimal

[4]Each state is a complete summary of the past history

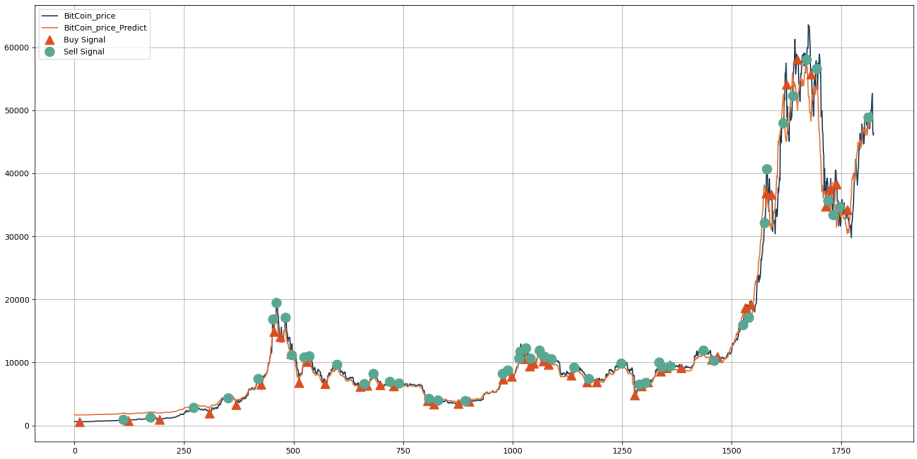

Figure 19: Bitcoins DP Result

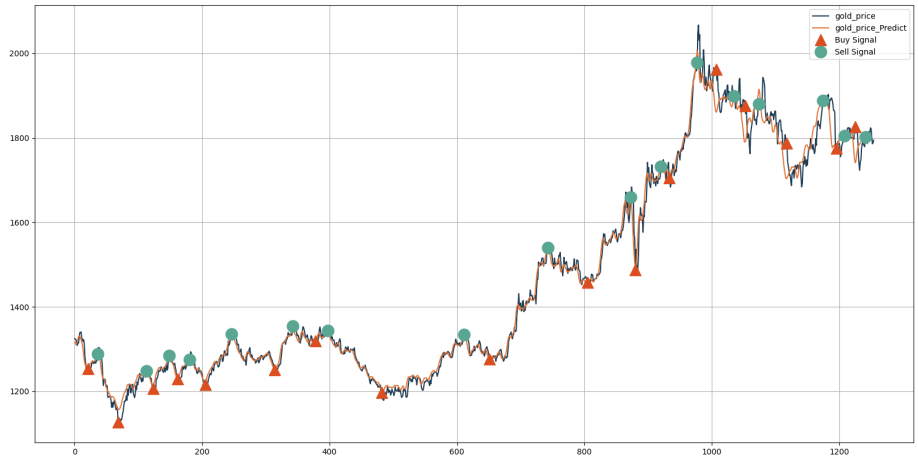

Figure 20: Gold DP Result

6 Sensitivity Analysis

6.1 Method

To test the sensitivity of transaction, we decide to test our model by changing the transaction fee rate from 50% to 150%. In other words, if the real Gold transaction fee rate is 1%, we will test how will the return rates change when we change the transaction fee rate from 0.5% to 1.5%. We will decide whether the model is sensitive to transaction cost according to the change of the final return.

6.2 Result

After making the transaction cost rate change from 50% to 150%, we create two plots, Figure 21, to show the influence of transaction fee rate on the final returns for these two stocks.

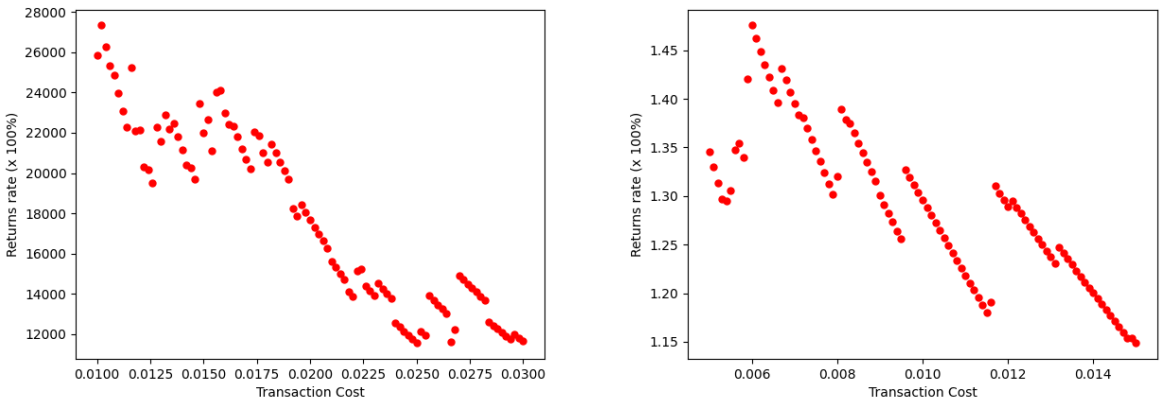

Figure 21: Sensitive Plot (Bitcoin / Gold)

From the figure above, the left one shows Bitcoins' change of return rate caused by increasing transaction cost, and the right one shows Gold's change of return rate caused by increasing transaction cost. The return rate change for Bitcoins changes from 26000 to 12000, decreasing for about 54%, while the return rate change for Gold changes from about 1.45 to 1.15, decreasing for about 21%. Although comparing the two stocks, Bitcoins is more sensitive to the change of transaction cost, the change for both stocks shows that they are all sensitive to the transaction cost. This result is caused mainly by the dynamic programming model. Since the dynamic programming model is to optimize the final return by catching each likely chance to trade. Thus, even if the change of transaction cost from 0.01 to 0.03 does not seem like a great number, it causes a great change to the final return.

7 Strengths and Weaknesses

7.1 Strengths

- The Optimal Portfolio Model can both fit for two types of volatile assets allocation but also more types of volatile assets allocation.
- Dynamic Programming Model can accurately predict each possible profitable chance to invest.

7.2 Weaknesses

- Our model results may be over-fitted, because the number of real data used for validating the model is too small.
- Our model might be simple and not realistic since it is only based on the closing price of Gold and Bitcoin without the highest and lowest daily price of these two volatile assets.

- The output from Dynamic Programming Model changes corresponding to change in transaction commission.

8 Conclusion

To predict the trend of Bitcoin and Gold, we choose time series models, ARIMA and LSTM. We perform parameter tuning and final result comparison for both ARIMA and LSTM models, respectively. In ARIMA, we also try to dynamize the model but give up this approach due to overfitting at last. Thus, we will choose the LSTM Model for prediction since the results of our LSTM model are much better than those of the ARIMA model, both in terms of visualization and MSE comparison. After accessing the predicted asset daily prices, we use Dynamic Programming to predict the optimal dates to trade and the type of asset to trade. After applying the prediction to the raw data, we appreciate the investment from $1000.00 to $14,140,234.70.

9 Memorandum

DATE: Jan 1, 2022
TO: Market Traders
FROM: Team 2222562
SUBJECT: Trading Strategies for Gold and Bitcoin

Let's think of the daily float and Bitcoin and Gold prices as stocks. Then for stocks, predicting the future value requires considering the previous values. So for Bitcoin and Gold price prediction, we are using the LSTM model that "remembers" the previous input values. The LSTM model is a neural network algorithm that stimulates neurons to arrive at a relatively more accurate model. Compared to other models, LSTM floats more significantly on a daily basis. This means that you should optimize short-term trading by frequently buying and selling volatile assets.

Before you make a decision on short-term trading, we recommend you first find out the best portfolio weight for investment. In our model, based on the predicted daily price data on Gold and Bitcoins over the recent five years, the results show the best weight is 0.188 : 0.812. That is to say, when you decide to use $1000 for beginning investment, the combination of the optimal weight should be allocating $188 on Gold and $812 on Bitcoin. One of our model advantages shows that we can get the optimal weight for different circumstances, even when choosing 20 different types of volatile assets.

After you decide on allocation weights on volatile assets, to better help with your short-term trading decision-making process, we develop a model using Dynamic Progressing Model to optimize every day's return rate to strive for the highest overall return rate. We can offer you the best timeline for trading different types of volatile assets through this method. Our application of this model on the Gold and Bitcoin successfully appreciates the investment from $1,000.00 to $14,140,234.70, which is a significant result in the trading market.

Strategies List:

1. You should choose several volatile assets based on balancing risks and benefits for hedging.
2. You can use our LSTM model to make predictions on the future daily price of your volatile assets based on our LSTM model.
3. You should use Optimal Portfolio Model to find out the best portfolio weights for investment.
4. You can use the trading timeline from the Dynamic Programming Model to trade assets on time.

References

[1] Taras Bodnar, Stepan Mazur, and Yarema Okhrin. Bayesian estimation of the global minimum variance portfolio. *European Journal of Operational Research*, 256(1):292–307, 2017.

[2] Rene D. Estember and Michael John R. Maraña. Forecasting of stock prices using brownian motion monte ..., Mar 2016.

[3] Xing Jin and Allen X. Zhang. Decomposition of optimal portfolio weight in a jump-diffusion model and its applications. *Review of Financial Studies*, 25(9):2877–2919, 2012.

[4] YAREMA OKHRIN and WOLFGANG SCHMID. Estimation of optimal portfolio weights. *International Journal of Theoretical and Applied Finance*, 11(03):249–276, 2008.

[5] Shay Palachy. Stationarity in time series analysis, Sep 2019.

[6] Samik Raychaudhuri. Introduction to monte carlo simulation. In *2008 Winter Simulation Conference*, pages 91–100, 2008.

[7] Jayesh Salvi. Significance of acf and pacf plots in time series analysis, Mar 2019.

Appendices

Appendix A: Part of our LSTM Source Code

```
# Slice the data into train and test sets
train <- dataset[1:train_size]
test <- dataset[(train_size + 1):length(dataset)]
look_back <- 5
trainXY <- create_dataset(train, look_back)
testXY <-  create_dataset(test, look_back)

dim_train <- dim(trainXY$dataX)
dim_test <- dim(testXY$dataX)

# Reshape input to be [samples, time steps, features]
dim(trainXY$dataX) <- c(dim_train[1], 1, dim_train[2])
dim(testXY$dataX) <- c(dim_test[1], 1, dim_test[2])

# Fit the Model
model <- keras_model_sequential()

# Train the model
trained_model <- model %>%
  layer_lstm(units = 10,
```

```
            input_shape = c(1, look_back)) %>%
  layer_dense(units = 1) %>%
  compile(loss = 'mean_squared_error',
          optimizer = 'adam') %>%
  fit(
    trainXY$dataX,
    trainXY$dataY,
    epochs = 20,
    batch_size = 97,
    verbose = 1,
    validation_split = 0.25
  )
plot(trained_model)

trainScore <- model %>%
    evaluate(
        trainXY$dataX,
        trainXY$dataY,
        verbose = 2)

testScore <- model %>%
    evaluate(
        testXY$dataX,
        testXY$dataY,
        verbose = 2)

sprintf(
    'Train␣Score:␣%.4f␣MSE␣(%.4f␣RMSE)',
    trainScore * spread^2,
    sqrt(trainScore) * spread)

sprintf(
    'Test␣Score:␣%.4f␣MSE␣(%.4f␣RMSE)',
    testScore * spread^2,
    sqrt(testScore) * spread)

trainPredict <- model %>%
  predict(trainXY$dataX,
          verbose = 1)
testPredict <- model %>%
  predict(testXY$dataX,
          verbose = 1)

trainPredict <- trainPredict * spread + min_value
testPredict <- testPredict * spread + min_value
```