Problem Chosen	2021 MCM/ICM Summary Sheet	Team Control Number
C		2222703

Market Trading Helper: Decision Strategy Based on Long-Short Term Memory and Classification Decision

Summary

With the development of economics, we pay more attentions on investment to make more money. Gold and Bitcoin market is important way to increase in the value of our property. How to make money in the markets become a significant issue, and it is worthy of taking time and effort to solve the problem. Based on the Gold and Bitcoin daily price in the five year, we hope to establish an automatic decision model to get more investment worth with initial $1000 though investing two kinds of assets by hold, purchase or sale. The details are as follows:

As for the question 1, based on the data of the Gold and Bitcoin price, we present a model to predicting the daily price though the past 30 days price by Long-Short Term Memory (LSTM) with gating mechanism, which is method of deep learning. The performance of predictor is good enough to make the trading decision, the predicting value matches the tendency of the real price changing of the Gold and Bitcoin markets. Then, we overcome the problem that Bitcoin can be traded every day, but Gold is only traded on days the market is open, a decision model is proposed with five 6 classification to decide the hold, purchase or sale of cash, Gold and Bitcoin. it's worth noting that we design parameters to stop loss and stop profit to avoid trading too frequently, it can realize tread long-term to make more profits. Different stop loss and stop profit parameters means different profits. By five years trading, the maximal initial state is about [0, 0, 66604].

As for the equation 2, in the decision model, we design two stop loss and stop profit parameters for Gold and Bitcoin, respectively, and we iterate over all possible and reasonable value of the parameters to get the maximal profit. Besides, we consider the influence of changing of the stop loss and stop profit parameters. Noteworthy, for almost the possible and reasonable stop loss and stop profit parameters, we profit or didn't lose in the five trading, it show the effectiveness of the automatic decision model. Thus, it is believe that the model provides the best strategy.

As for the equation 3, we choose four different stop loss and stop profit parameters to analyze sensitivity of transaction costs by iterating over all the possible and reasonable transaction costs from 1% to 10%. We find that as the transaction costs increase, the profit will reduce, and the profit reduce 0%, 10%, 27%, 64.5%, respectively. However, we always profit or didn't lose.

As for the equation 4,we write a memorandum to describe our model, and give some suggestion for traders.

In conclusion, we overcome the problems that Gold and Bitcoin can not be traded at the same time in some days, and we design the stop loss and stop profit parameters to make the model suitable for long-term trading. The model include a predictor based on LSTM and decision equation with five classification. We believe that our model can provides the best trading strategy, and we analyze the sensitivity of transaction costs.

Key words: Deep learning, Predictor based on LSTM, Classification Decision, Stop loss and stop profit parameters, Long-term trading

Contents

1 Introduction **1**
1.1 Problem Restatement . . . 1
1.2 Literature Review . . . 1
1.3 Data Processing . . . 2
1.4 Modeling Framework . . . 2

2 Nomenclature **2**

3 Predictor model **3**
3.1 Model building . . . 3
3.2 Model Results . . . 4

4 Classification Decision **5**
4.1 Data Reconstruction . . . 5
4.1.1 Trading Decision . . . 6
4.1.2 Account changing following the Decision 1-6 . . . 7
4.2 Decision Results . . . 8

5 Sensitivity Analysis and stability analysis **10**

6 Strengths and Weaknesses **10**
6.1 Strengths . . . 10
6.2 Weaknesses . . . 11

7 Conclusion **11**

8 Memorandum **13**

9 References **15**

10 Appendices **15**

1 Introduction

Generally speaking, devaluation may occur in the real world, people may buy some stocks to hedge, but the risk also exists! In practice,there are a mount of the past trade data, but the future data is unknown. The future data may have connection with the past data, digging the relationship between them may be benefit for trading. The past data is time sequential, and is very big. Therefore, it needs some special methods to deal with.

1.1 Problem Restatement

In this report, we focuses on the Gold and Bitcoin price data in five-year trading period, the trading has been made to buying or selling both of Gold and Bitcoin, one of them or doing nothing to make more money at low risk. But " Bitcoin can be traded every day, but Gold is only traded on days the market is open".

To gain more profits at low risk, detailedly, we need to

- Find the relationship of the past data and future data,how to according to the relationship realize the trading strategy.
- How to many the large and robust profit by the proposed model.
- Find the relationship of transaction costs and profit.

1.2 Literature Review

In the early predict analysis have proposed several methods for this problem. In 2006,[1] ,Grey Prediction, and In 2000,[2]Learning to forget: Continual prediction with LSTM But due to the limited time, we finally chose the LSTM predictor model to solve it which has the highest prediction accuracy.Thin about the part of classification decision,we proposed the methods for the decision. In 2000,[3]A greedy algorithm for aligning DNA sequences and In1972,[4] Goal programming for decision analysis of multiple objectives.through thinking we decision combine the multiple objectives programming and the Greedy Algorithm.We also look at the literature in 2016, [5]Bitcoin: Economics, technology, and governance. Journal of economic Perspectives in order to analyzing the Bitcoin Market. The strengths and weaknesses of the planning is show in Figure 1

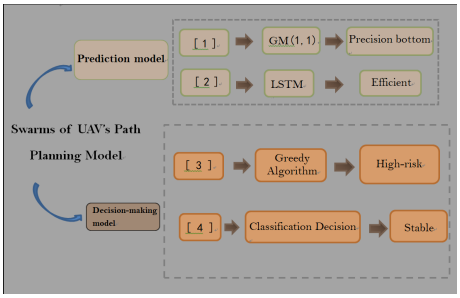

Figure 1: strengths and weaknesses of the planning

1.3 Data Processing

we find the data of Gold have many missing value. In the predictor model we delete the missing value, and in the classification decision the miss value be filled in, the detail will be introduced in hereinafter.And than we standardize the data to avoid vanishing gradient and exploding gradient during model construction,and eliminate the influence of the dimension.

1.4 Modeling Framework

The modeling framework can be presented as shown in Figure 2.

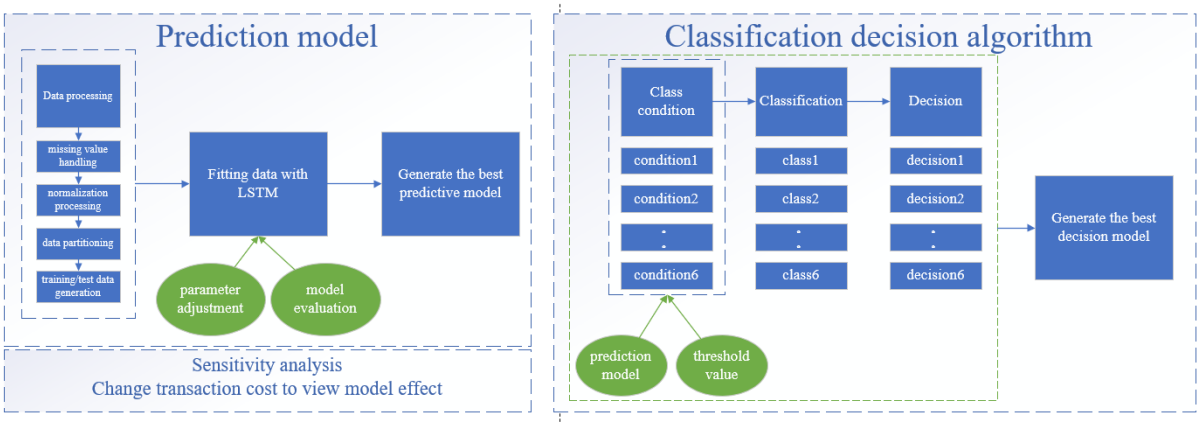

Figure 2: Modeling Framework

2 Nomenclature

The importance notations in this report is as follows:

Table 1: Notations

Symbol	Description
$[C(i), G(i), B(i)]$	cash, Gold, and Bitcoin worth in In the account of the i-th day
x_t	t-th Gold price in the data of deleting missing values
y_t	t-th Bitcoin price in the data
$\hat{x}_{t+1}$	$t+1$-th predicting value of Gold price in the data of deleting missing values
$\hat{y}_{t+1}$	$t+1$-th predicting value Bitcoin price in the data
$X, Y, \hat{X}, \hat{Y}$	vectors
$X(i), Y(i), \hat{X}(i), \hat{Y}(i)$	i-th element of those vectors
η	yield rate
ξ	one-dimensional variable
β_g	stop loss and stop profit parameter of Gold
β_b	stop loss and stop profit parameter of Bitcoin

3 Predictor model

3.1 Model building

As there are some missing value in the Gold data, first of all, we delete the missing value. Let the x_t denotes the t-th Gold price in the data of deleting missing values, y_t means t-th Bitcoin price in the data, the $\hat{x}_{t+1}$ denotes $(t+1)$-th predicting value of Gold price in the data of deleting missing values, and $\hat{y}_{t+1}$ is $(t+1)$-th predicting value Bitcoin price in the data. Taking Gold data as example, we want to establish the relationship between predicting value $\hat{x}_{t+1}$ and real past values $x_t, , x_{t-1}, \cdots$. Suppose that the prediction of the $(t+1)$-th value $\hat{x}_t$ has relationship with the past 30 real values $x_t, , x_{t-1}, \cdots, x_{t-29}$. Thus our target is to find a function

$$\hat{x}_{t+1} = F(x_t, x_{t-1}, \cdots, x_{t-29}), \tag{1}$$

to make accumulated error between $\hat{x}_{t+1}$ and x_{t+1} for all t small enough. Accordingly , for the Bitcoin, we also want to find a function

$$\hat{y}_{t+1} = G(y_t, y_{t-1}, \cdots, y_{t-29}), \tag{2}$$

to make accumulated error between $\hat{y}_{t+1}$ and y_{t+1} for all t small enough.

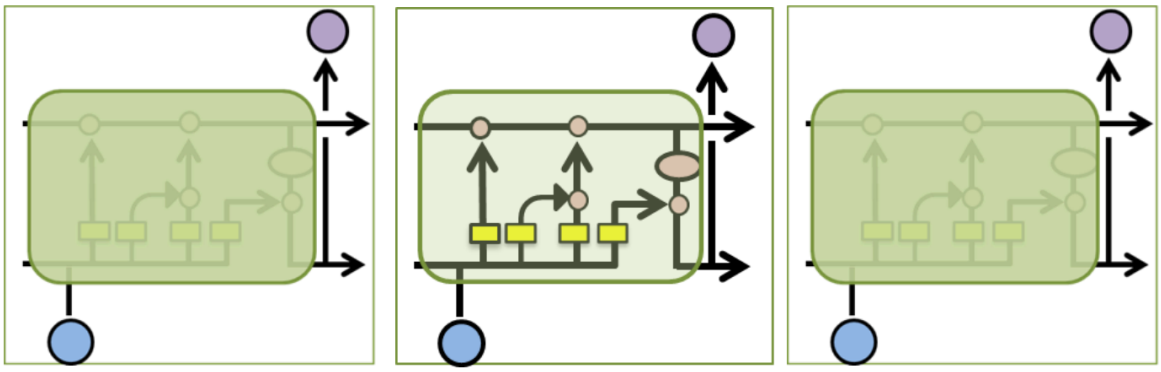

Figure 3: LSTM Network

To find the predictor (1) and (2), the LSTM model is used. The frame of LSTM model is shown in Figure 3. LSTM cell, see figure 4, is implement as follows:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}c_{t-1} + b_i),\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f),\\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c),\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o),\\
h_t &= o_t \tanh(c_t).
\end{aligned} \tag{3}$$

There are three kinds of gates in a LSTM cell, that is input gate, output gate and forget gate. In this report, we use three layers LSTM network, see figure 3. LSTM network are same as RNNS, except the hidden layer updates are substituted by LSTM cell. For the sequence tagging task. The LSTM network is very useful.

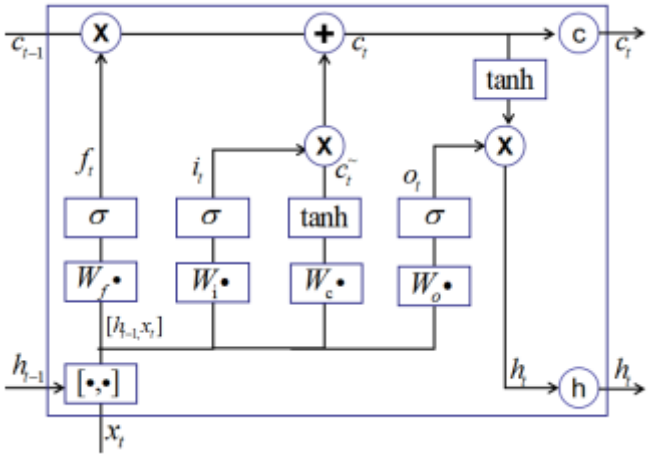

Figure 4: LSTM cell.

3.2 Model Results

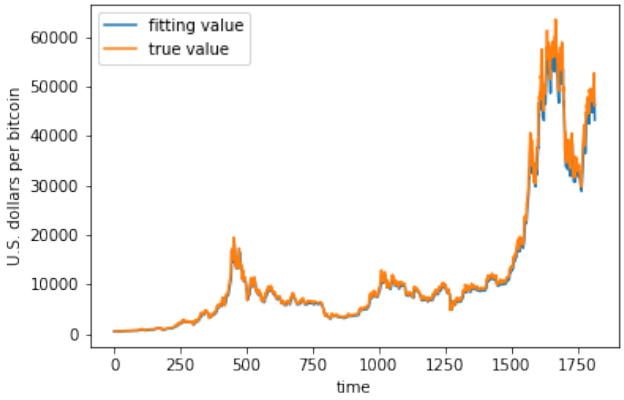

Figure 5: Time series of predicted value and real value of Gold.

We separate the data as training set and test set, the size of train set for Gold and Bitcoin is 900 and 1300, respectively, the rest is for test set. By using LSTM network, the daily price of the Gold and Bitcoin is predicted by the past 30 days real data. Figure 5 shows the time series of predicted value and real value of Gold, the prediction seems well, the prediction matches the tendency of gold price changing. Figure 6 shows predicting performance of Bitcoin, it seems well in tendency. However, the $RMSE$ (Root Mean Square Error) in the test set are not very small, the $RMSE$ for gold price prediction is about 107, the $RMSE$ for Bitcoin price prediction is about 24866. Thus we guess that:

It is difficult to predict the future price of Gold and Bitcoin precisely, but its tendency may be can predicted very well. Thus, it is very risky to trading the stocks frequently due to difficult to predict the future price precisely, and we should follow the tendency to trading the Gold and Bitcoin.

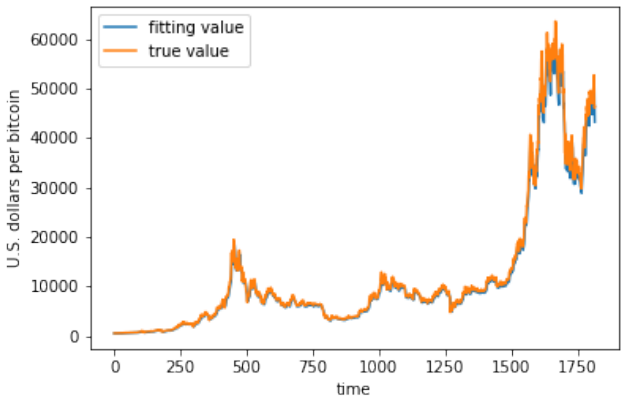

Figure 6: Time series of predicted value and real value of Bitcoin.

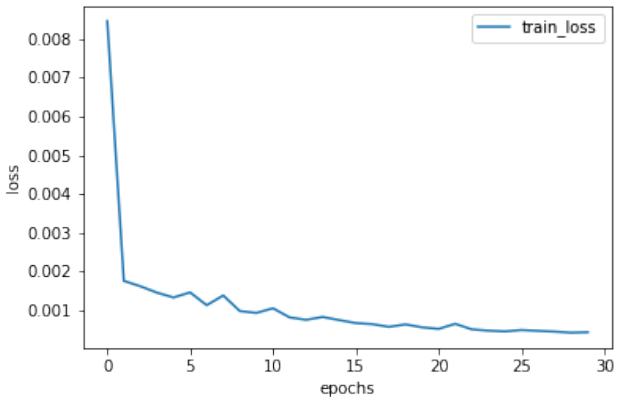

Figure 7: Train loss by Bitcoin with normalization

In the figures 7 and 8, we give the train loss of the model with different epochs for gold and Bitcoin after normalization. It shows that the model is very well.

4 Classification Decision

4.1 Data Reconstruction

We rebuilt following date, the $X(i)$ denotes the Gold price of the i-th day, if there is no price in that day, for example the Gold trading is closed, then the value of the day is missing. The $Y(i)$ denotes the Bitcoin price of the i-th day. The $\hat{X}(i)$ denotes the prediction of Gold price the $(i+1)$-th day by the predictor (1), if doesn't has, then missing. The $\hat{Y}(i)$ denotes the prediction of Bitcoin price of the $(i+1)$-th day by the predictor (2).

Handling missing values: if $X(i)$ is missing, then $X(i) := X(i-1)$, and $\hat{X}(i) := X(i-1)$.

The missing values handling is very important in this report, on one hand, the Gold and Bitcoin can not be traded meantime for some days, there must has some missing value in Gold data for these days,

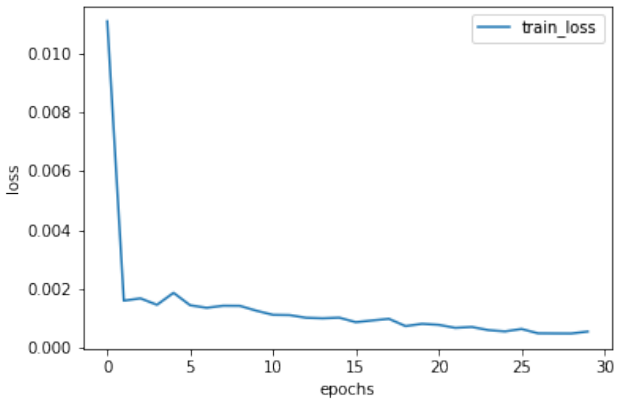

Figure 8: Train loss by Gold with normalization

and the Gold data has 10 missing value inherently. Besides, the handling method of missing values influence the final decision, the following decision is presented based on the missing values handling.

if the i-th day is Gold missing day (must be $\hat{X}(i) = X(i)$), then we don't trade the Gold in the i-th day, besides the Gold market may not open in that day. Thus the decisions are as follows:

4.1.1 Trading Decision

Decision 1: If

$$\begin{aligned} &\hat{X}(i+1) = X(i+1), \\ &\frac{\hat{Y}(i+1) - Y(i+1)}{Y(i+1)} > \beta_b\% \end{aligned} \tag{4}$$

then the Gold doesn't traded, and the prediction yield rate of $(i+1)$-th day is greater than the stop loss and stop profit parameter of Bitcoin, thus the trading decision of the $(i+1)$-th day is:

Gold hold, Cash → Bitcoin.

Decision 2: If

$$\begin{aligned} &\hat{X}(i+1) = X(i+1), \\ &\frac{\hat{Y}(i+1) - Y(i+1)}{Y(i+1)} < -\beta_b\% \end{aligned} \tag{5}$$

then the Gold doesn't traded, and the prediction yield rate of $(i+1)$-th day is less than the stop loss and stop profit parameter of Bitcoin, thus the trading decision of the $(i+1)$-th day is:

Gold hold, Bitcoin → Cash.

Decision 3: If

$$\begin{aligned} &-\beta_b \leq \frac{\hat{Y}(i+1) - Y(i+1)}{Y(i+1)} \leq \beta_b, \\ &\frac{\hat{Y}(i+1) - Y(i+1)}{Y(i+1)} < -\beta_b\% \end{aligned} \tag{6}$$

then the loss and profit is less than the stop loss and stop profit parameter , thus the trading decision of the $(i+1)$-th day is:

Cash hold, Gold hold, Bitcoin hold.

Decision 4: If

$$\begin{aligned} &\frac{\hat{Y}(i+1)-Y(i+1)}{Y(i+1)} \leq \frac{\hat{X}(i+1)-X(i+1)}{X(i+1)}, \\ &\frac{\hat{X}(i+1)-X(i+1)}{X(i+1)} > \beta_g\% \end{aligned} \tag{7}$$

the trading decision of the $(i+1)$-th day is:

Cash and Bitcoin → Gold.

Decision 5: If

$$\begin{aligned} &\frac{\hat{Y}(i+1)-Y(i+1)}{Y(i+1)} \geq \frac{\hat{X}(i+1)-X(i+1)}{X(i+1)}, \\ &\frac{\hat{Y}(i+1)-Y(i+1)}{Y(i+1)} > \beta_b \end{aligned} \tag{8}$$

the trading decision of the $(i+1)$-th day is:

Cash and Gold → Bitcoin.

Decision 6: If

$$\begin{aligned} &\frac{\hat{Y}(i+1)-Y(i+1)}{Y(i+1)} < -\beta_b, \\ &\frac{\hat{X}(i+1)-X(i+1)}{X(i+1)} < -\beta_g \end{aligned} \tag{9}$$

the trading decision of the $(i+1)$-th day is:

Gold and Bitcoin → Cash.

In above decision, we consider that the Gold and Bitcoin may can not be traded in same days by the Decisions 1-3. Besides, we use the stop loss and stop profit parameters to avoid the cases of trading frequently, and realize the long-term trading. In the next, the account changing will be proposed based on above Decisions.

4.1.2 Account changing following the Decision 1-6

If the account in the i-th day is $[C(i), G(i), B(i)]$, the account in the $(i+1)$-th day is presented as follow:

For the Decision 1, the in the $(i+1)$-th day is

$$\begin{aligned} &C(i+1) = 0, \\ &G(i+1) = G(i), \\ &B(i+1) = B(i)\left(1+\frac{Y(i+1)-Y(i)}{Y(i)}\right) + C(i)(1-\alpha_b\%). \end{aligned} \tag{10}$$

For the Decision 2, the in the $(i+1)$-th day is

$$\begin{aligned} C(i+1) &= B(i)\left(1+\frac{Y(i+1)-Y(i)}{Y(i)}\right)(1-\alpha_b\%)+C(i),\\ G(i+1) &= G(i),\\ B(i+1) &= 0. \end{aligned} \tag{11}$$

For the Decision 3, the in the $(i+1)$-th day is

$$\begin{aligned} C(i+1) &= C(i),\\ G(i+1) &= G(i)\left(1+\frac{X(i+1)-X(i)}{X(i)}\right),\\ B(i+1) &= B(i)\left(1+\frac{Y(i+1)-Y(i)}{Y(i)}\right). \end{aligned} \tag{12}$$

For the Decision 4, the in the $(i+1)$-th day is

$$\begin{aligned} C(i+1) &= 0,\\ G(i+1) &= G(i)\left(1+\frac{X(i+1)-X(i)}{X(i)}\right)+B(i)\left(1+\frac{Y(i+1)-Y(i)}{Y(i)}\right)(1-\alpha_g\%-\alpha_b\%),\\ B(i+1) &= 0. \end{aligned} \tag{13}$$

For the Decision 5, the in the $(i+1)$-th day is

$$\begin{aligned} C(i+1) &= 0,\\ G(i+1) &= 0,\\ B(i+1) &= B(i)\left(1+\frac{Y(i+1)-Y(i)}{Y(i)}\right)+C(i)(1-\alpha_b\%)+G(i)\left(1+\frac{X(i+1)-X(i)}{X(i)}\right)(1-\alpha_g\%-\alpha_b\%). \end{aligned} \tag{14}$$

For the Decision 6, the in the $(i+1)$-th day is

$$\begin{aligned} C(i+1) &= C(i)+B(i)\left(1+\frac{Y(i+1)-Y(i)}{Y(i)}\right)(1-\alpha_b\%)+G(i)\left(1+\frac{X(i+1)-X(i)}{X(i)}\right)(1-\alpha_g\%),\\ G(i+1) &= 0,\\ B(i+1) &= 0. \end{aligned} \tag{15}$$

4.2 Decision Results

Based on subsections 4.1.1 and 4.1.2, the account changing as the day goes can be calculated in Python. The Time series of the account reflects the real trading results, since the Gold and Bitcoin cannot be traded meantime for some day has been considered and we do not use the future data in the Predictor.

The value of the stop loss and stop profit parameters of Gold and Bitcoin has impact on the profit, simulation results with different β_g and β_b has been made, as shown in figures 9 and 10. In the figure 9, the step size of β_g, β_b is 1, the arrange of β_g is from 4 to 100, and the arrange of β_b is from 6

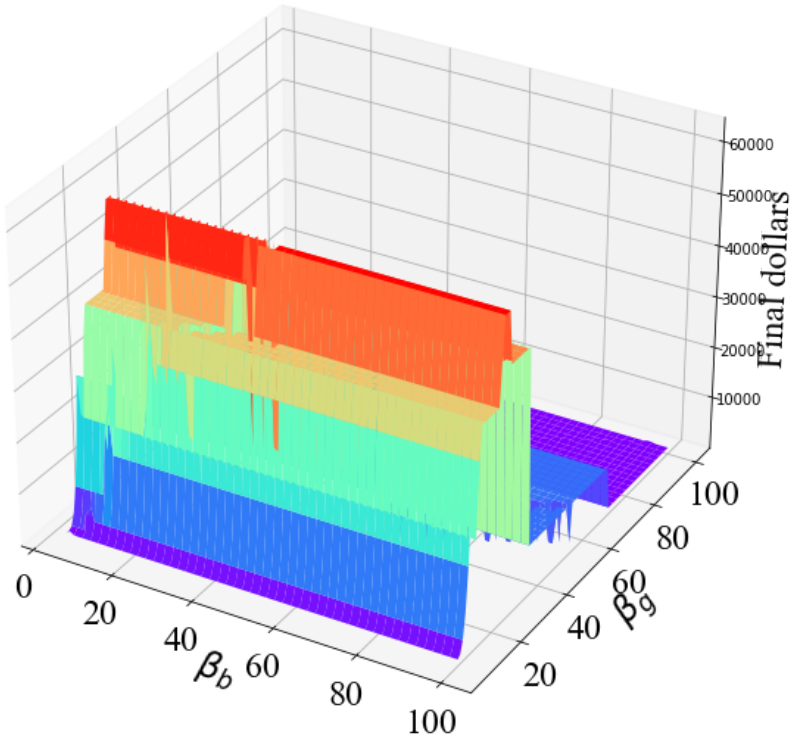

Figure 9: The final profit with variables β_g, β_b

to 100. The maximal value of the account is $[0, 0, 66604.95]$, where $\beta_b = 17$ and $\beta_g = 5$. In the figure 10, we calculation the possibility of profit for β_g, β_b with uniform distribution in the region $4 \leq \beta_g \leq 100, 6 \leq \beta_b \leq 100$ by

$$P(\eta \geq \xi) \approx n(\eta \geq \xi) \tag{16}$$

where n denotes the frequency in the simulation result, and

$$\eta = \frac{C(final) + G(final) + B(final) - 100}{100} \tag{17}$$

denotes the rate of return in the five years.

In the two figures we find that **there is not loss in our decision model, the profit is abundant in the most time**, which shows the effectiveness of the decision model.

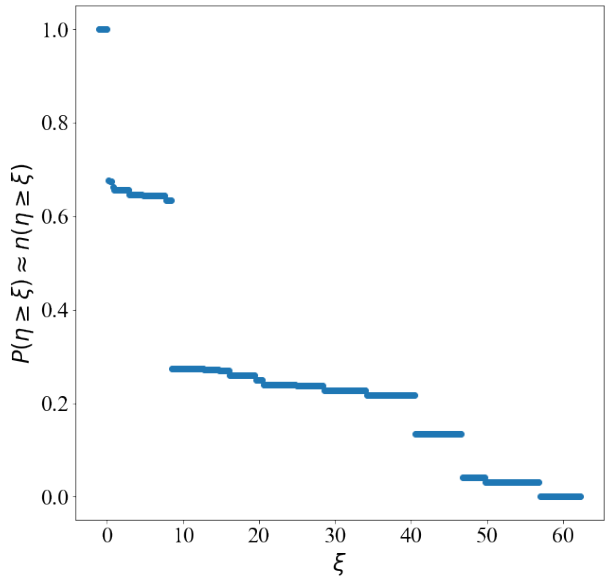

Figure 10: Possibility of the final profit.

5 Sensitivity Analysis and stability analysis

In this section, we rearrange the account $C(final)+G(final)+B(final)$ from small to large, and find the 0%, 25%, 50%, 100% values of the account, the corresponding $[\beta_g, \beta_b]$ are $[28, 95]$ and $[11, 49]$, $[93, 13]$ and $[5, 17]$, respectively. We use the four data to analyze the sensitivity and stability of transaction costs α_g and α_b.

The transaction costs α_g, α_b range from 1 to 10, the overlarge transaction costs are meaningless. The result is shown in figures 11 and 12. In the figure 11, the final account $C(final)+G(final)+B(final)$ is changed as the independent variables α_g and α_b, and in figure 12, we suppose that $\alpha_g = \alpha_b$, where the step size is 0.1.

From the figures, the final account $C(final) + G(final) + B(final)$ decrease when the α_g, α_b increase. The maximal losses for different α_g and α_b are 0%, 10%, 27% and 64.5% for the four points. But, anyway, **there is no loss in the four cases, and the scale of decrease seems can be acceptable.**

6 Strengths and Weaknesses

6.1 Strengths

- **Automatic trading helper:** We propose the predictor model based on LSTM network, and a classification decision model, which can make the trading decision automatically in different stocks.

- **Effectiveness in portfolio investment:** Our model design overcomes the problem that some stocks can be traded meantime for some day. It is very useful for portfolio investment.

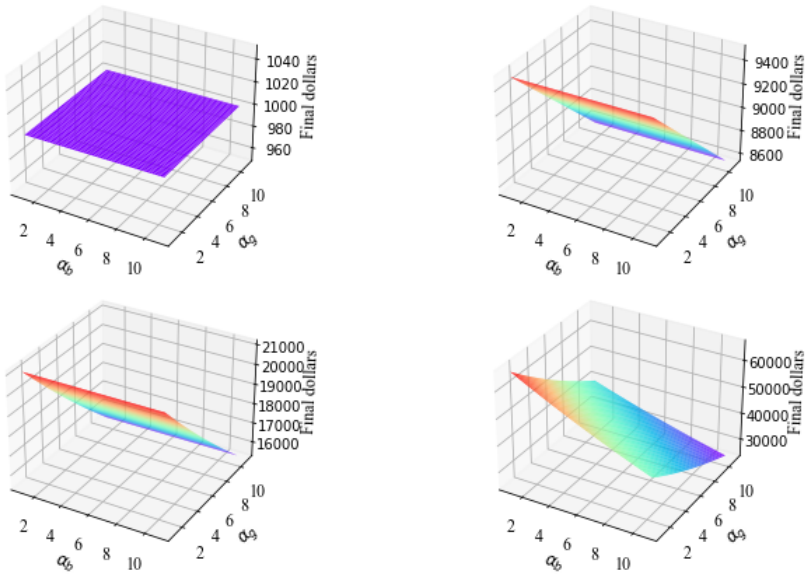

Figure 11: The final account $C(final) + G(final) + B(final)$ with different α_g and α_b.

- **Robustness:** In our model, there is no loss under the reasonable parameter. In the most time, we make a pretty penny.
- **Long-term trading:** Our model can realize the long-term trading by designing stop loss and stop profit parameters, which is simple

6.2 Weaknesses

- **Limitations of our acquired data set:** Because of the limitations of the problem we can only train our model with the given data.

7 Conclusion

In our work, we propose the LSTM predictor model and classification decision to provide accurately predict and making efficient decision. By using the past data, we give the prediction of the future data, a design a classification decision model, which overcome the Gold and Bitcoin can not be traded at the same time for some day. The biggest highlights may be that we realize the long-term trading, which are more grounded in reality. Besides, our model has robustness on the changing of transaction cost. All the models in this report are realized by Python, and can be recurred.

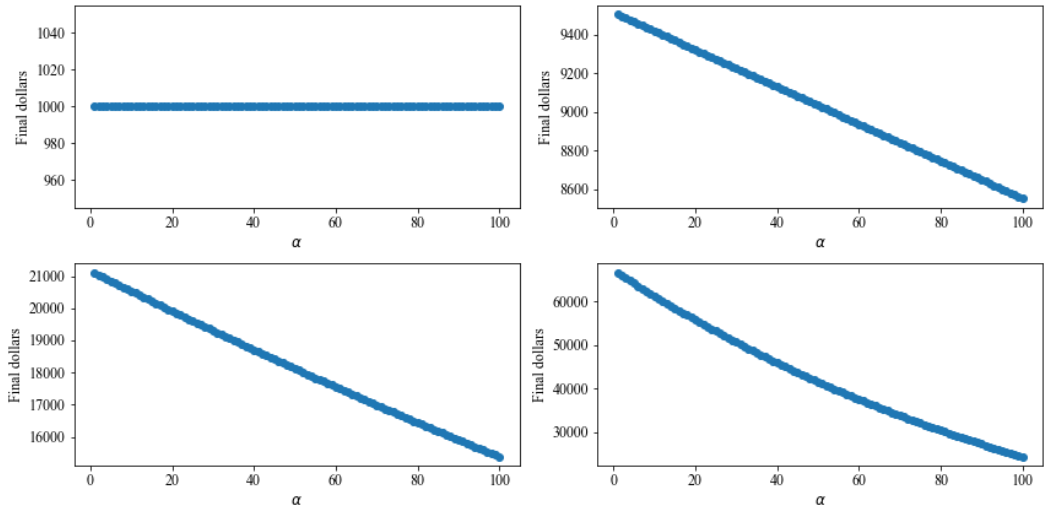

Figure 12: The final account $C(final) + G(final) + B(final)$ with different $\alpha_g = \alpha_b$.

8 Memorandum

From: Team# 2222703

To: Trader

Data February 21,2022

Subject: On Market Trading Model

Dear Sir or Madam:

It is our great honor to provide you our model with reference value of financial products investment strategy. Based on the daily price of gold and Bitcoin in the data files provided in previous years, we build mathematical models to determine the tendency of the price changing in future, and design a trading prediction system with high precision, small difference and accurate results for you. Our methods are as follows.

First, We conduct statistical analysis by collecting daily prices of gold and Bitcoin over a period of time. In order to accurately analyze the daily movements of gold and Bitcoin , we built a model to predict the pattern of the next day's price of the product, and a decision model has been proposed with 6 classifications to help the Gold and Bitcoin trading.The predictor plus the classifications decision model can give the suggestions whether the financial products will be traded in the future, Finally, we verify the sensitivity stability of the models, which show our model can make a pretty penny, and even the transaction cost is large, we also can make money.

For short-term financial products such as gold Bitcoin, the have the characteristics of low threshold, flexible trading and high returns. Therefore, we seize its characteristics and choose to trade before observing product price changes in order to seek maximum benefits. Here are some conclusions and suggestions based on our findings.

- we select a period of time in the past transactions were analyzed, because the gold trading time, lead to loss of data, so we will fill data, again normalized processing, then the data fitting, get volumes over a period of change model, but the data is super, we made several attempts to obtain reasonable results, Get the use of a few days before the trading volume of the next day to adjust the trading volume. Finally, the data is fitted and the prediction model is obtained through multiple-layers training.

- The predictor is the first step to realize automatic trading in Gold and Bitcoin. There more importance is to make a decision. We give a 6 classifications decision model to realize portfolio, which overcome the disadvantage that different stocks has different transaction time, which is a kinds of automatic trading helper.

- When making decisions, it is very important to judge timing of trading. In order to avoid large losses, we also set artificial thresholds to help us judge the timing of trading. When there is a short and small amount of loss or profit, we observe in advance whether the total cost of the replacement of investment products exceeds the threshold, to choose whether to replace the trading products, so as to avoid large losses and frequent transaction fee losses caused by the sudden change of transaction amount, and to put the trading perspective in the long run and seek greater interests.

To sum up, we propose the following suggestions: Due to the large scale of the investment market, the overall investment stability and the demand for information exchange in the adjustment process are

large, so more transaction volume data are needed to support the accuracy of the model. Investment behavior should not be dispersed during investment, because the stability of investment is poor, so it is necessary to judge the proportion of investment through model prediction and choose a more stable plan. This model has strong market response ability and better processing ability for risk control.

9 References

References

[1] Liu S, Lin Y. Grey Prediction[M]. Springer London, 2006.

[2] Gers F A, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM[J]. Neural computation, 2000, 12(10): 2451-2471.

[3] Zhang Z, Schwartz S, Wagner L, et al. A greedy algorithm for aligning DNA sequences[J]. Journal of Computational biology, 2000, 7(1-2): 203-214.

[4] Klahr C N. Multiple objectives in mathematical programming[J]. Operations research, 1958, 6(6): 849-855.

[5] Urquhart A. The inefficiency of Bitcoin[J]. Economics Letters, 2016, 148: 80-82.

[6] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016.

[7] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural networks, 2005, 18(5-6): 602-610.

[8] Bang-Jensen J, Gutin G, Yeo A. When the greedy algorithm fails[J]. Discrete optimization, 2004, 1(2): 121-127.

10 Appendices

Gold prediction model

```
from datetime import date
import pandas as pd
import quandl
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from tensorflow.keras import layers, Sequential
import tensorflow as tf

gpus = tf.config.experimental.list_physical_devices('GPU')
print(gpus)
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

start = date(2000,10,12)
end = date.today()
google_stock = pd.read_csv("./LBMA-GOLD.csv").dropna(axis=0)

def run(time_stamp = 30):
    global google_stock
    google_stock = google_stock[['Value']]
```

```
    train = google_stock[0:900 + time_stamp]
    valid = google_stock[900 + time_stamp:]

    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(train)
    x_train, y_train = [], []

    for i in range(time_stamp, len(train)):
        x_train.append(scaled_data[i - time_stamp:i])
        y_train.append(scaled_data[i, 0])

    x_train, y_train = np.array(x_train), np.array(y_train)

    scaled_data = scaler.fit_transform(valid)
    x_valid, y_valid = [], []
    for i in range(time_stamp, len(valid)):
        x_valid.append(scaled_data[i - time_stamp:i])
        y_valid.append(scaled_data[i, 0])

    x_valid, y_valid = np.array(x_valid), np.array(y_valid)

    scaled_data = scaler.fit_transform(google_stock)
    x_total, y_total = [], []
    for i in range(time_stamp, len(google_stock)):
        x_total.append(scaled_data[i - time_stamp:i])
        y_total.append(scaled_data[i, 0])

    x_total, y_total = np.array(x_total), np.array(y_total)

    epochs = 30
    batch_size = 16
    model = Sequential()
    model.add(layers.LSTM(units=100, return_sequences=True,
              input_dim=x_train.shape[-1], input_length=x_train.shape[1]))
    model.add(layers.LSTM(units=50))
    model.add(layers.Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1)


    x_valid_predict = model.predict(x_valid)
    scaler.fit_transform(pd.DataFrame(valid['Value'].values))
    x_valid_predict = scaler.inverse_transform(x_valid_predict)
    y_valid = scaler.inverse_transform([y_valid])

    rms = np.sqrt(np.mean(np.power((y_valid - x_valid_predict), 2)))
    print(f"test_rms:{rms}")
    return rms
run()
```

Bitcoin prediction model

```
from datetime import date
```

```
import pandas as pd
import quandl
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from tensorflow.keras import layers, Sequential
import tensorflow as tf

gpus = tf.config.experimental.list_physical_devices('GPU')
print(gpus)
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

start = date(2000,10,12)
end = date.today()
google_stock = pd.read_csv("./BCHAIN-MKPRU.csv").dropna(axis=0)

def run(time_stamp = 30):
    global google_stock
    google_stock = google_stock[['Value']]

    train = google_stock[0:1200 + time_stamp]
    valid = google_stock[1200 + time_stamp:]

    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(train)
    x_train, y_train = [], []

    for i in range(time_stamp, len(train)):
        x_train.append(scaled_data[i - time_stamp:i])
        y_train.append(scaled_data[i, 0])

    x_train, y_train = np.array(x_train), np.array(y_train)

    scaled_data = scaler.fit_transform(valid)
    x_valid, y_valid = [], []
    for i in range(time_stamp, len(valid)):
        x_valid.append(scaled_data[i - time_stamp:i])
        y_valid.append(scaled_data[i, 0])

    x_valid, y_valid = np.array(x_valid), np.array(y_valid)

    scaled_data = scaler.fit_transform(google_stock)
    x_total, y_total = [], []
    for i in range(time_stamp, len(google_stock)):
        x_total.append(scaled_data[i - time_stamp:i])
        y_total.append(scaled_data[i, 0])

    x_total, y_total = np.array(x_total), np.array(y_total)

    epochs = 30
    batch_size = 16
    model = Sequential()
    model.add(layers.LSTM(units=100, return_sequences=True,
```

```
                input_dim=x_train.shape[-1], input_length=x_train.shape[1]))
    model.add(layers.LSTM(units=50))
    model.add(layers.Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1)


    x_valid_predict = model.predict(x_valid)
    scaler.fit_transform(pd.DataFrame(valid['Value'].values))
    x_valid_predict = scaler.inverse_transform(x_valid_predict)
    y_valid = scaler.inverse_transform([y_valid])

    rms = np.sqrt(np.mean(np.power((y_valid - x_valid_predict), 2)))
    print(f"test_rms:{rms}")
    return rms
run()
```

Choice of decision model

```
from datetime import date
import pandas as pd
import quandl
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from tensorflow.keras import layers, Sequential,models
import tensorflow as tf
import pickle

df = pd.read_csv("./input.csv")
xt = list(df["usd"])
xt_1 = list(df["usd_nextday_predict"])
yt = list(df["Value"])
yt_1 = list(df["BCHAIN-MKPRU_nextday_predict"])
beta_c = 8  # 1-100
beta_b =15 #1-100

for i in range(len(xt)-1):
    if xt[i+1] == 0:
        xt[i+1] = xt[i]
        if xt_1[i+1] == 0:
            xt_1[i+1] = xt[i+1]

def run(erfa_b = 2,erfa_c = 1):
    result = [1000,0,0] # crash gold bitcion
    for i in range(len(xt)-1):
        if xt[i+1]==xt_1[i+1] and
           yt_1[i+1]>(1+beta_b*0.01)*yt[i+1]:
            result[2]=result[2]*(1+(yt[i+1]-yt[i])/yt[i])+\
                      result[0]*(1-erfa_b*0.01)
            result[0]=0
        elif xt[i+1]==xt_1[i+1] and
             yt_1[i+1] < yt[i+1]*(1-beta_b*0.01):
            result[0]=result[0]+result[2]*
```

```
                    (1+(yt[i+1]-yt[i])/yt[i])*(1-erfa_b*0.01)
            result[2]=0
        elif xt[i+1]*(1-beta_c*0.01)<=xt_1[i+1]<=(1+beta_c*0.01)*xt[i+1] and
             yt[i+1]*(1-beta_b*0.01)<=yt_1[i+1]<=(1+beta_b*0.01)*yt[i+1]:
            result[1]=result[1]*(1+(xt[i+1]-xt[i])/xt[i])
            result[2]=result[2]*(1+(yt[i+1]-yt[i])/yt[i])

        elif xt_1[i+1]>(1+beta_c*0.01)*xt[i+1] and
            (xt_1[i+1]-xt[i+1])*yt[i+1]>=(yt_1[i+1]-yt[i+1])*xt[i+1]:
            result[1]=result[1]*(1+(xt[i+1]-xt[i])/xt[i]) + \
                      result[2]*(1+(yt[i+1]-yt[i])/yt[i])*
                      (1-erfa_b*0.01-erfa_c*0.01) +\
                      result[0]*(1-erfa_c*0.01)
            result[0] = 0
            result[2] = 0
        elif yt_1[i+1] > (1+beta_b*0.01)*yt[i+1] and
             (yt_1[i+1]-yt[i+1])*xt[i+1] >= (xt_1[i+1]-xt[i+1])*yt[i+1]:
            result[2] = result[2]*(1+(yt[i+1]-yt[i])/yt[i]) + \
                        result[0]*(1-erfa_b*0.01) + \
                        result[1]*(1+(xt[i+1]-xt[i])/xt[i])*
                        (1-erfa_b*0.01-erfa_c*0.01)
            result[0] = 0
            result[1] = 0
        elif xt_1[i+1] < xt[i+1]*(1-beta_c*0.01) and
             yt_1[i+1] < yt[i+1]*(1-beta_b*0.01):
            result[0] = result[0] + \
                        result[1]*(1+(xt[i+1]-xt[i])/xt[i])*(1-erfa_b*0.01)+\
                        result[2]*(1+(yt[i+1]-yt[i])/yt[i])*(1-erfa_c*0.01)
            result[1] = 0
            result[2] = 0

    return result

def main():
    global beta_b
    global beta_c
    test = dict()
    for i in range(6, 100+1):
        beta_b = i
        for j in range(4, 100+1):
            beta_c = j
            test[str((i,j))] = sum(run())
            print(test[str((i,j))])
    with open("result.pkl","wb") as f:
        pickle.dump(test,f)

if __name__ == '__main__':
    main()
```

sensitivity analysis

```
from datetime import date
import pandas as pd
import quandl
```

```
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from tensorflow.keras import layers, Sequential, models
import tensorflow as tf
import pickle

df = pd.read_csv("./input.csv")
xt = list(df["usd"])
xt_1 = list(df["usd_nextday_predict"])
yt = list(df["Value"])
yt_1 = list(df["BCHAIN-MKPRU_nextday_predict"])
beta_b = 95  # 1-100
beta_c = 28  # 1-100

for i in range(len(xt) - 1):
    if xt[i + 1] == 0:
        xt[i + 1] = xt[i]
        if xt_1[i + 1] == 0:
            xt_1[i + 1] = xt[i + 1]

def run(erfa_b = 2,erfa_c = 1):
    result = [1000,0,0] # crash gold bitcion
    for i in range(len(xt)-1):
        if xt[i+1]==xt_1[i+1] and
           yt_1[i+1]>(1+beta_b*0.01)*yt[i+1]:
            result[2]=result[2]*(1+(yt[i+1]-yt[i])/yt[i])+\
                        result[0]*(1-erfa_b*0.01)
            result[0]=0
        elif xt[i+1]==xt_1[i+1] and
             yt_1[i+1] < yt[i+1]*(1-beta_b*0.01):
            result[0]=result[0]+result[2]*
                      (1+(yt[i+1]-yt[i])/yt[i])*(1-erfa_b*0.01)
            result[2]=0
        elif xt[i+1]*(1-beta_c*0.01)<=xt_1[i+1]<=(1+beta_c*0.01)*xt[i+1] and
             yt[i+1]*(1-beta_b*0.01)<=yt_1[i+1]<=(1+beta_b*0.01)*yt[i+1]:
            result[1]=result[1]*(1+(xt[i+1]-xt[i])/xt[i])
            result[2]=result[2]*(1+(yt[i+1]-yt[i])/yt[i])

        elif xt_1[i+1]>(1+beta_c*0.01)*xt[i+1] and
            (xt_1[i+1]-xt[i+1])*yt[i+1]>=(yt_1[i+1]-yt[i+1])*xt[i+1]:
            result[1]=result[1]*(1+(xt[i+1]-xt[i])/xt[i]) + \
                      result[2]*(1+(yt[i+1]-yt[i])/yt[i])*
                      (1-erfa_b*0.01-erfa_c*0.01) +\
                      result[0]*(1-erfa_c*0.01)
            result[0] = 0
            result[2] = 0
        elif yt_1[i+1] > (1+beta_b*0.01)*yt[i+1] and
             (yt_1[i+1]-yt[i+1])*xt[i+1] >= (xt_1[i+1]-xt[i+1])*yt[i+1]:
            result[2] = result[2]*(1+(yt[i+1]-yt[i])/yt[i]) + \
                        result[0]*(1-erfa_b*0.01) + \
                        result[1]*(1+(xt[i+1]-xt[i])/xt[i])*
                        (1-erfa_b*0.01-erfa_c*0.01)
            result[0] = 0
```

```
            result[1] = 0
        elif xt_1[i+1] < xt[i+1]*(1-beta_c*0.01) and
             yt_1[i+1] < yt[i+1]*(1-beta_b*0.01):
            result[0] = result[0] + \
                        result[1]*(1+(xt[i+1]-xt[i])/xt[i])*(1-erfa_b*0.01)+\
                        result[2]*(1+(yt[i+1]-yt[i])/yt[i])*(1-erfa_c*0.01)
            result[1] = 0
            result[2] = 0

    return result

def main():
    temp = [(95, 28),(49, 11),(13, 93),(17, 5)]
    for k in range(4):
        global beta_b
        global beta_c
        beta_b, beta_c = temp[k]
        data = dict()
        for i in np.arange(1,11,0.1):
            for j in np.arange(1,11,0.1):
                data[str((i, j))] = sum(run(i,j))
                print(data[str((i, j))])
        with open(f"result{k}.pkl","wb") as f:
            pickle.dump(data,f)


if __name__ == '__main__':
    main()
```