Stock system introduction

A stock simulation system developed for successful university machine learning and bioinformatics courses. The system has stock market data for Taiwan stocks from 2004 to now, and users need to implement a forecast model through these raw materials. The system executes the user's model every day and decides how to invest in the stock tomorrow based on the forecast. Each person's total assets will be drawn into a line chart according to time and compared with other people. After logging in, you can check the operation details such as purchase and sale. The process of using this system can be roughly divided into the following steps:

1. Reading data
2. Use model
3. Generate decision
4. Submit program

The details of each step are explained below

1.Reading data

In luffy's(server) /home/mlb/res/stock/twse/json/ directory, the number, opening price, closing price, highest and lowest price, and volume of each stock are stored. The format is as follows:

In [1]:

```
stock = {
  "0050": {
      "adj_close": "84.90",
      "close": "84.95",
      "high": "85.80",
      "low": "84.90",
      "open": "85.80",
      "volume": "20996200"
  },
  "0051" : {
      "adj_close": "32.50",
      "close": "32.50",
      "high": "33.20",
      "low": "32.46",
      "open": "33.20",
      "volume": "92075"
  },
  "0052" : {
      "adj_close": "54.30",
      "close": "54.50",
      "high": "54.90",
```

```
    "low": "54.50",
    "open": "54.90",
    "volume": "102150"
  }
}

print(stock['0050'])
print(stock['0050']['close'])
```

{'adj_close': '84.90', 'high': '85.80', 'volume': '20996200', 'low': '84.90', 'open': '85.80', 'close': '84.95'}
84.95

2. Use model

After reading the stock data, the way to construct the model can be referred to the teaching here. When using the model, you must develop a program that can read the specified time, stock data generation characteristics (such as the acquisition of all stocks in the week before the forecast, and is characterized by the opening price, closing price and average price), and These features are entered into the model that has been trained to obtain the predicted results.

3. Generate decision

The system saves the investment decision into a json file called Decision File. The content includes information about which stock to buy, how much to buy, and how long to hold. Students must develop a program that converts the results of the predictive model into a Decision File. Usually the forecast results only include the stock's ups and downs (if using a classification model) or the expected stock price (if a regression model is used), unless you build multiple models to determine each parameter in the Decision File, otherwise you have to buy and hold How long the parameters, etc., require the user to write additional conditions to judge. The format of Decision File is as follows

In [ ]:

```
[ # an array, each element is an object

  {
    "code": "1214", # Stock code
```

```
    "type": "buy", # two operations, buy (buy) / short (short)
    "open_price": 16.75, # is equal to or less than this price to buy, the transaction will happen


    # each person has 1 unit of virtual cash at the beginning; the cash available for investment changes
    # When investing every day, the system will calculate that each person can spend one unit of money per day.
    # When investing every day, the system will recalculate the ownership and convert it into a percentage.
    # Suppose only buy two stocks, 1214 weight is 1,2351, weight is 3, and the cost after calculation is 1/4 and 3/4 respectively, multiplied by the cash available on the day.
    # Take this example as an example, buy 1214 with 1/4; vent 2351 with 2/4

    "weight": 1,

    # One of the following three conditions will be sold
    "close_high_price": 16.8, # Sell higher than this price
    "close_low_price": 16.3 # Selling at a lower price than this price (stop loss point)
    "life": 5, # If neither of the above conditions occurs, it will be sold at the closing price in 5 days.

  },
  {
    "code": "2351",
    "life": 3,
    "type": "short",
    "weigth": 3,
    "open_price": 27.3,
    "close_high_price": 27.5,
    "close_low_price": 25.8
  }
]

# Note that Decision File cannot contain annotations when actually working.
# Note that this program section is not a python code and cannot be executed inside a jupyter notebook.
```

Note that if there is no Decision File on a certain day or the content is empty, the system will automatically invest all the cash in the market (TAIEX, 0050), encouraging users to invest more. Based on past experience, following a large-cap investment will be worse than predicting the performance of the model.

4. Submit program

Each user has a stock folder under the luffy account. The structure is:

- ~/stock/bin Program directory, read only
- ~/stock/commit The main output directory for storing Decision File
- ~/stock/output Alternate output directory for temporary archives

The program required by the above process is placed in the ~/stock/bin directory to complete the submission, and the system will fetch the program at a specific time each day. The following is a sample process. The detailed content and name can be changed arbitrarily. All the temporary file suggestions in the process are placed in the ~/stock/output directory:

1. Reading data: Develop a load.py program to prepare the data. Because the date is different every day, this program usually needs the date of the day as a parameter.
2. Use the model to develop a predict.py program to use the model to predict the results of the day.
3. Generate Decisions: Develop a make_decision.py program that generates a Decision File based on the predicted results and additional conditions. The final Decision File must be placed in the ~/stock/commit directory. The file name format is YYYY-MM-DD_YYYY_MM_DD.json. The date is the forecast date. For example, the Decision File location of 2018-09-10 should be ~/stock/commit/ 2018-09-10_2018-09-10.json.

Finally, the ~/stock/bin directory must also have a run.sh program (file names cannot be changed). Run.sh is a shell script, which must contain the entire process of predicting stocks. When run.sh is executed successfully, ~/stock/commit/ should generate the Decision File for the day. When the system executes run.sh, it will add two parameters at the end, representing the start and end dates of the forecast, such as sh run.sh 2018-09-10 2018-09-10. These two parameters can be obtained in shell script with $1 and $2. The following is an example of run.sh:

In [ ]:

```
# if command is `sh run.sh 2018-09-10 2018-09-10`

# generate the data to predict
./load.py $1 5 ../output/data

# predict with a trained model
./predict.py ../output/data ../output/pred

# make decision
./make_decision.py ../output/pred ../commit/$1_$2.json

# the output file MUST be ../commit/2018-09-10_2018-09-10.json
# note that the code of this cell is python, please do not run this cell
```

- Remarks
  When the system is actually running, the ~/stock/bin/ directory will be copied to a separate virtual machine, and the corresponding ../commit and ../output empty directories will be created, and run.sh will be executed, so it cannot be saved during execution. Take the file other than ~/stock/bin/.
- The model should be pre-trained and placed directly in ~/stock/bin/. When the system executes run.sh for more than one minute, it will be automatically interrupted, so if run.sh contains model training, it usually exceeds the time limit, resulting in the failure to generate Decision File.
- The run-test function of the system web page can test whether your current run.sh can successfully generate Decision File.
- The sim-test of the system webpage provides simulation test. The temporary archive can be seen on the luffy, which is convenient for debugging. However, due to the large number of courses, the function of stock-sim is turned off, and the prediction and adjustment of the model are treated as actual simulations. reference.
-