JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 39, 1101-1116 (2023)
DOI: 10.6688/JISE.202309_39(5).0006

Design and Implement an Intelligent System for Stock Investment Decision Making*

CHIUNG HON LEE[1], TSUNG-JU LEE[2], YU-SHENG CHIANG[1],
SHIH-YI YUAN[3] AND JYH-HWA LIOU[2]
[1]*Department of Information Engineering and Computer Science*
[2]*Department of Finance*
[3]*Department of Communications Engineering*
Feng Chia University
Taichung, 407 Taiwan
E-mail: {cholee@; freeman.tjlee@mail; d0636094@mail; syyuan@; jhliou@}fcu.edu.tw

A machine learning (ML) based software for stock investment decision making is designed and implemented to explore problems of developing the financial software with intelligent capabilities. Two main issues are discussed in this paper: how to integrate the process of software development and ML module development; how to integrate the ML modules into the software. A utility optimization problem is proposed to formulate software design considerations. In the prototype system, three modules are implemented to facilitate the investment decision making process: a fundamental analysis module; a stock chip analysis module; and a technical analysis module. Those modules let the user to sieve candidate stocks for investment and help the user to judge whether or not it's a good timing to invest. For making better user experience, we implement a user interface in a social communication software.

Keywords: software process, machine learning software engineering, financial software, investment decision making, utility optimization problem

1. INTRODUCTION

One of the trends of the financial software industry is around how to integrate machine learning capabilities into the software [1, 2]. It's obvious that integrating artificial intelligence capabilities into a financial software will enhance its functions of predictive or data analysis, *etc.* However, the process of developing the machine learning (ML) module is different from traditional software development process. How to integrate the machine learning module development and software development process when develop an intelligent financial system become an important issue [3].

Following the rapidly development of ML technology, it has been widely used in different software applications. In [4], the author presents an industrial case study on how to apply ML to automatically detect transaction errors and propose corrections in a business software system. A survey of how software teams develop, deploy and maintain software solutions that involve ML components has been identified in [5]. The paper concludes 6 classes of ML software engineering best practices. We also refer to those practices to check our ML software development. In [6], some software design patterns for ML applications has been proposed.

Received October 31, 2022; accepted December 12, 2022.
Communicated by Lok Ka Man.
* This research was partially supported by the National Science and Technology Council, Taiwan under grant MOST 111-2410-H-035 -011-.

A machine learning process and a basic software development process are shown in Fig. 1. In the machine learning process, some steps are data oriented such as data collection, data clearing, and data labeling. Some steps are model oriented such as feature engineering, model training, and model evaluation, etc. What is the dependency among those steps and the software development process and when to deploy the trained ML model to the software are important issues of machine learning software engineering. The software for different industry has different characteristics and might use different ML capabilities. In this paper, we focus on how to design and implement an intelligent software system to help stock investors to make their investment decisions. We use this case study to explore the process of machine learning software engineering in stock investment field.

In our system, three modules are implemented to sieve and evaluate stocks. The chip analysis module analyze the shareholders distribution for filtering the target which the most market leader bought and many small retail investors sold; A fundamental analysis module is designed to evaluates a stock is worth to invest or not by its financial statements such as monthly revenue, quarterly report, earnings per share (EPS), *etc.*; The technical analysis module based on the technical analysis such as stock technical indexes or candlestick patterns identification [7] to suggest investor whether or not it's a good time point to buy or sell stocks.

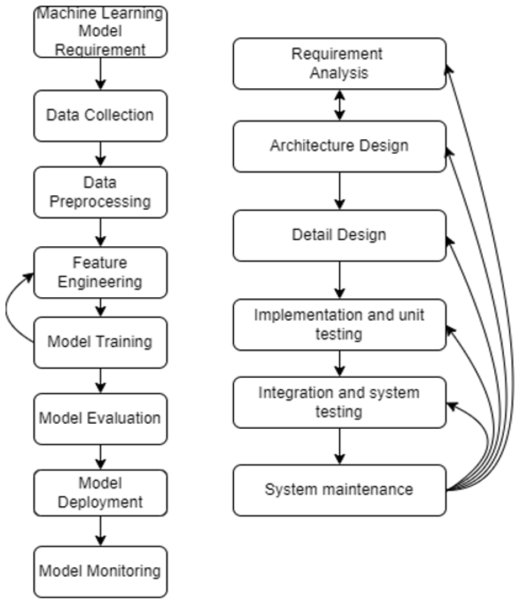
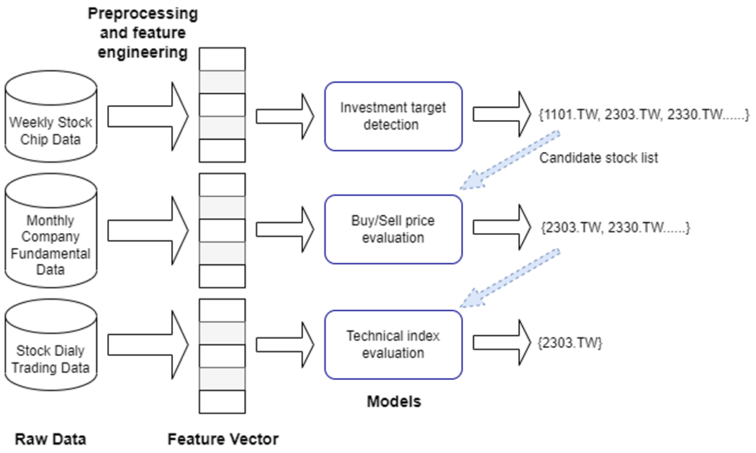


Fig. 1. A software and a machine learning development process.

The rest of this paper is organized as follows. In next section, the investment decision making software architecture are introduced. The system development process and a prototype implementation are illustrated in Section 3. Section 4 discusses the dependency between machine learning and software development process to explore issues of the machine learning software engineering in stock investment domain. Finally, conclusions and future works are given in the Section 5.

2. INVESTMENT DECISION MAKING ARCHITECTURE

The system architecture of investment decision making to demonstrate our investment idea is shown in Fig. 2. In [8], the research empirical result shows that the change of stock share-holders' number is related to stock investment returns in a significant level. We used this result to help the design our investment decision making processes.


Preprocessing and feature engineering
Weekly Stock Chip Data
Monthly Company Fundamental Data
Stock Dialy Trading Data
Investment target detection
Buy/Sell price evaluation
Technical index evaluation
{1101.TW, 2303.TW, 2330.TW......}
Candidate stock list
{2303.TW, 2330.TW......}
{2303.TW}
Models
Raw Data
Feature Vector
Fig. 2. The investment decision making solution architecture.

There are three processes to find out a stocks list for investment. First step is using shareholder distribution data to filter candidate stocks which the market leader interested in. Second step is to check the stock fundamental information of those candidate stocks and get a recommend stock list. The last step is use technical analysis results to generate buy or sell signals. The processes of ML modules design are: data collecting, data preprocessing and feature engineering, model training, and model evaluation.

There are three different kind of data collected for the system: weekly shareholder data, monthly company fundamental data, and stock daily trading data. Because the data announced in different period and have different source, we used the practices proposed in [5] to help our development. For example, using sanity checks for all external data sources or checking that input data is complete, balanced and well distributed.

Because the original raw data have many features, we have to check which feature is related to the result which we want. An example of shareholder distribution table used in first step is shown in Table 1. Which is announced by Taiwan Stock Exchange every week [9]. The data shown in table is TSMC (2330) which announced in 20 May 2022.

Table 1. An example of shareholder distribution table.

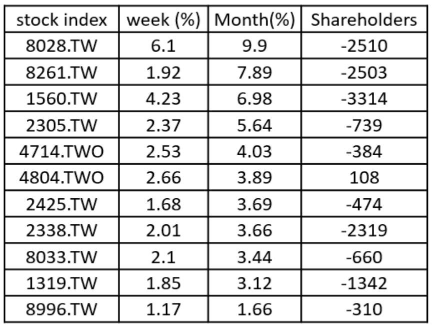
	Unit grading	Number of people	Number of units	Percentage (%)
1	1-999	826659	143477359	0.55
2	1000-5000	47431	817955930	3.15
3	5001-10000	15520	344834226	1.32
…	…	…	…	…
15	>10000001	1517	22442470247	86.54

Because the investor can get the stock unit distribution variation information by comparing the shareholder distribution table in different weeks, the table can be used to understand how much stock unit bought by the market leader (hold more than 1000,000 stock units) or how much new investors enter the market to buy the stock in this week. Based in the research result in [8], we use those data to obtain features used in our chip analysis module. The basic idea is if a stock has been brought a lot by the market leader and many retail investors sold the stock, the stock price might rise. If the market leader sold a lot and many retail investors brought the stock, the stock price might fall. Because not every stock fit the proposed idea, we trained a ML model to check the relationship between the change of shareholders' number and the stock price variation.

Table 2 shows a real preprocessed data of the shareholder distribution of some stocks at 27 May 2022 in our system. The first column is stock index. Second and third columns stored percentage of the stock brought by market leader weekly and monthly. The last column reports amount variation of shareholders. The market leader is defined as follows:

If the stock price <100 NT dollars,
    market leader = share holder holds more than 1000 lots
If the stock price ≥ 100 NT dollars,
    market leader= share holder holds more than 400 lots

Table 2. Data of shareholders distribution.



stock index	week (%)	Month(%)	Shareholders
8028.TW	6.1	9.9	-2510
8261.TW	1.92	7.89	-2503
1560.TW	4.23	6.98	-3314
2305.TW	2.37	5.64	-739
4714.TWO	2.53	4.03	-384
4804.TWO	2.66	3.89	108
2425.TW	1.68	3.69	-474
2338.TW	2.01	3.66	-2319
8033.TW	2.1	3.44	-660
1319.TW	1.85	3.12	-1342
8996.TW	1.17	1.66	-310

As the table shown, we can find that the market leader brought 6.1% and 9.9% of stock 8028.TW weekly and monthly respectively and there are 2510 shareholders sold out their stocks from 23 May 2022 to 27 May 2022. We assume that the market leader knows more information about the stock than retail investors. If the market leader buy and retail investor sell a stock, then we can reason that the stock might have more chance to rise than other stocks.

The stock fundamental data used in our system are monthly revenue, quarterly report, earnings per share (EPS), *etc.* We use those data to evaluate an appropriate investing price of the stock. If the price of the stock is lower than the evaluated price then it is a good candidate to invest.

The technical analysis module uses technical indexes to make invest signals. The stock daily trading data include open, close, high, and low prices and the trading volume. Based on those raw data, there are many stock technical analysis concepts and techniques have been established for stock investment such as Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Fast and Slow Stochastic (KD-line), and candlestick patterns [9], *etc*. Those stock price analysis tools are the basis when design some investment decision making modules.

Because most stocks have ex-rights and ex-dividends every year, the first step of data preprocessing is to adjust trading prices. The definition of the rate of return Y_i is shown as follows:

$$Y_i = \left(\frac{Adj\ Close_{i+20}}{Adj\ Close_i} - 1 \right) * 100.$$

Where *Adj Close* is the adjusted close price and *i* is the trading date.

There are more than 150 stock technical indexes used in the stock market for trend analysis. Those indexes can be considered as features to train the technical analysis ML model. However, using all stock technical indexes as features to train ML model is unnecessary and time consuming. Because it's well performance and widely used in financial field, the XGBoost [10] has been chosen as the ML model for technical analysis module.

We train a XGBoost model and use it's feature importance sequence to select candidate features. In our experiment, there are about 100 features have been chosen from 164 indexes for different stocks. After a forward selection strategy, there are about 50 features have been chosen to reach better prediction performance. The stocks listed in Taiwan 50 index are used in ours experiment and the training data is from 2012 to 2022. The experiment results are show in Table 3. The best test mean-square error (MSE) is 0.56 and most MSE are between 2.0-5.0. The ML steps for technical analysis module creation are shown in Fig. 3.

Table 3. Feature number in different stocks.

Stock_id	Train_R2	Train_MSE	Test_R2	Test_MSE	Feature_num
2330.TW	0.975943	1.074758	0.889538	4.744289	52
2317.TW	0.983973	0.749991	0.883728	5.920238	55
2308.TW	0.984565	0.864437	0.889628	5.535407	52
2412.TW	0.981991	0.098126	0.888642	0.560352	55
1303.TW	0.98577	0.53496	0.892685	3.500549	55
...	...	...	...	...	...
0050.TW	0.988453	0.235833	0.890649	2.548319	50

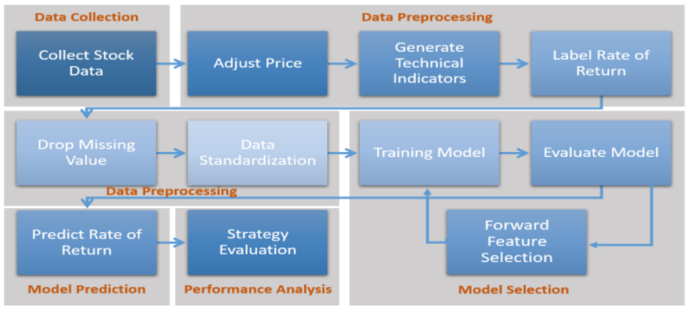


Fig. 3. The process for ML model creation.

3. SOFTWARE DEVELOPMENT PROCESS

In this section, we introduce the system development process of an intelligent stock investment system. The main function of the system is to help the stock investor to sieve and filter candidate stocks to invest via fundamental, shareholder distribution, and technical analysis. The stock sieving process is made by the return results from the ML modules.

3.1 Preliminaries

Based on the requirements acquisition results, there are three roles identified in the requirement analysis phase: investor, data analyzer, and data maintainer. Investor is the main user of the system; Data analyzer use the system to build ML models; Data maintainer use the system to collect and update stock data. A use case diagram is shown in Fig. 4 to represent functional requirements of the system. The use cases of the data maintainer are related to the data collecting process; The use cases of the data analyzer are related to data preprocessing and model-oriented ML processes.

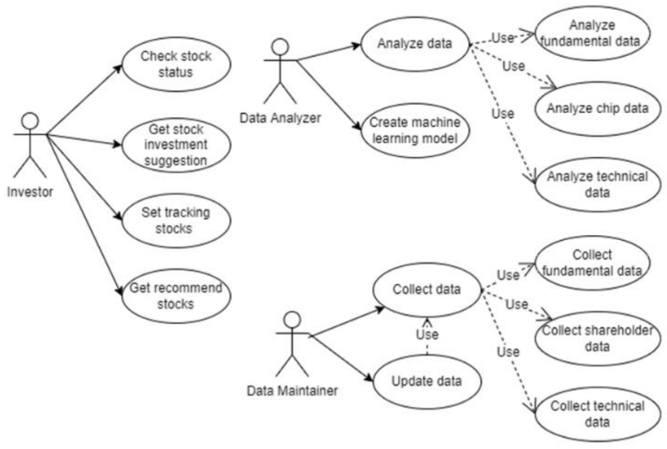


Fig. 4. Use case diagram of the system.

After interact with senior stock investors, we acquired four functional requirements to help investors to make their stock investment decisions. First use case is to check a specific stock status. When an investor is interested in a stock, this function can show basic information of the stock such daily trading data or fundamental data *etc.* Second use case indicates the scenario of an investor want to get suggestions from ML modules to check whether a stock worth to invest or not. Third use case shows the scenario that the user to set or delete stock indexes for tracking. If a stock on the tracking list, it's daily trading data will be reported automatically in every trading day. The last use case describes the scenario of getting a recommend stock list from ML modules. The use case of "get stock investment suggestion" and "get recommend stocks" are related to the ML models in our design.

The steps to get candidate stocks worth to invest are:

1. analyzing the shareholder distribution table to get candidate stock lists,
2. checking the fundamental data of those candidate stocks, and
3. generating buy/sell suggestion for stocks listed on the candidate stocks list to the investor.

3.2 System Design

Because the issue discussed in this paper is how to integrate the process of software development and machine learning (ML) module development, we only show the design of machine learning related use cases. A boundary, control, and entity analysis of the use case "get recommend stocks" and "get stock investment suggestion" are shown in Fig. 5. A sequential diagram of "get recommend stocks" use case is shown in Fig. 6.

A stock recommend interface is designed to interact with the investor. The "select candidate stocks" class is used to sieve candidate stocks from the return results of the "check stock status" class. The "check stock status" work with ML modules to do chip, fundamental, and technical analysis of stocks.

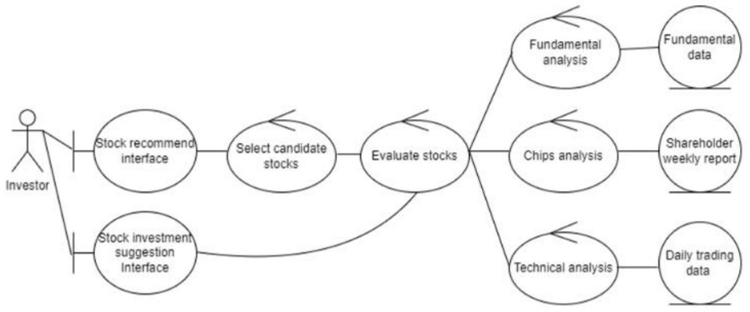


Fig. 5. A boundary, control, entity analysis of requirement.

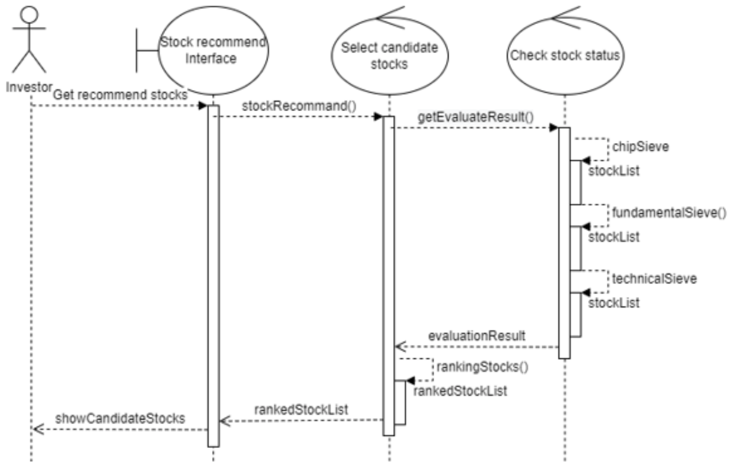
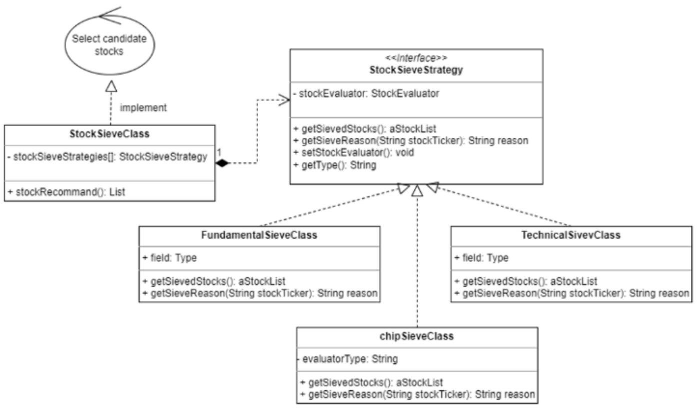


Fig. 6. Sequential diagram of "get recommend stocks" requirement.

Considering the system extension and maintenance requirements, we use strategy design pattern to design the stock sieve class [11]. After the "StockSieveStrategy" interface has been designed, different ML module designer can implement this interface to design their strategy to do chip, fundamental, and technical analysis. The design is shown in Fig. 7. If the stock sieve strategy has been changed in the future, the designer can change fundamental, chip, or technical sieve classes in a more flexible way.

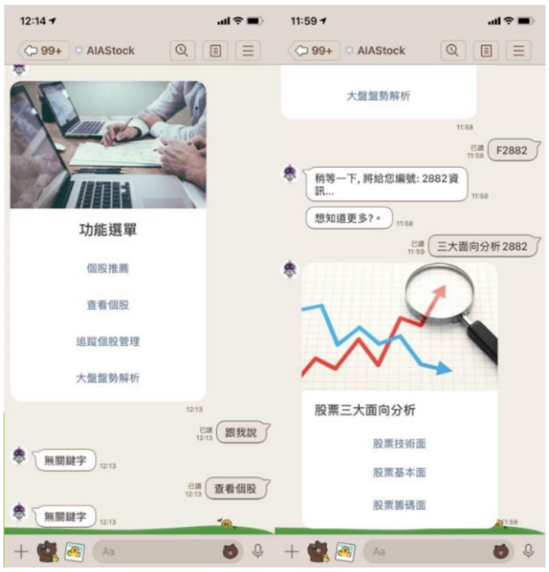

Select candidate stocks
implement
StockSieveClass
- stockSieveStrategies[]: StockSieveStrategy
+ stockRecommand(): List
1
<<interface>>
StockSieveStrategy
- stockEvaluator: StockEvaluator
+ getSievedStocks(): aStockList
+ getSieveReason(String stockTicker): String reason
+ setStockEvaluator(): void
+ getType(): String
FundamentalSieveClass
+ field: Type
+ getSievedStocks(): aStockList
+ getSieveReason(String stockTicker): String reason
TechnicalSivevClass
+ field: Type
+ getSievedStocks(): aStockList
+ getSieveReason(String stockTicker): String reason
chipSieveClass
- evaluatorType: String
+ getSievedStocks(): aStockList
+ getSieveReason(String stockTicker): String reason
Fig. 7. An implementation of "select candidate stocks" control class.

3.3 Implementation

A system prototype is design and implemented for demonstration. The programming language, tools, and third-party services used in our prototype are shown below:

- Programming language: Python
- Social communication interface: Linebot API
- Cloud platform: Heroku
- Database: mangoDB
- Stock data API: yfinance, yahoo_fin
- Message broadcast API: schedule

The user interface is shown in Fig. 8.



12:13
已讀 12:13 跟我說
無關鍵字 12:13
已讀 12:13 查看個股
無關鍵字 12:13

11:58
已讀 11:58 F2882
稍等一下, 將給您編號: 2882資訊... 11:58
想知道更多? 11:58
已讀 11:59 三大面向分析 2882
11:59

Fig. 8. The user interface of the system prototype.

4. DISCUSSIONS

Following the software development process discussed in Section 3, we explore the training data and ML model used in the ML modules in different aspects. There are three data types: fundamental, shareholder distribution, and technical data. Considering the characteristics of stock data, those different data types have different data source and updating cycle.

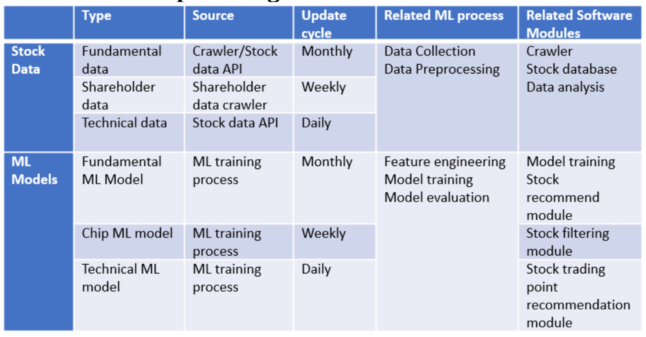
Because the trading data of stock market is changing every trading time and the revenue of a listed company is updating every month, the stock data collecting and processing have to be done in different time cycle. The data maintainer use implemented crawler or stock data API to maintain those data. Since the data has been updated, the data analyzer can use the system to re-train ML models. The data processing process and model training process can be done atomically or manually.

4.1 ML Process and Software Development Process Analysis

The relationships among different types of data and models in different aspects are shown in Table 4. We can get the relationships between ML process and software development process by tracking the development steps of data collecting, preprocessing, model training, and model evaluation modules.

The stock data crawler, stock database, and data analysis modules are designed to handle issues of stock data collection and data preprocessing. The model training, stock recommend, stock filtering, and stock trading point recommendation modules is related to the issues of ML model training and application. The stock data is collected by the crawler of stock data APIs, stored in a stock database, and processed by the data analysis module. The processed data will be the training data source of the model training modules and those trained and evaluated models is the basis of stock recommend, stock filtering, and stock trading point recommendation modules.

Table 4. The relationships among data and ML models in different aspects.

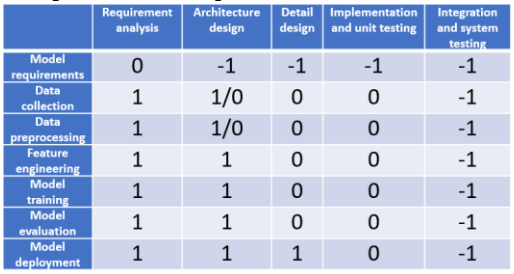


	Type	Source	Update cycle	Related ML process	Related Software Modules
Stock Data	Fundamental data	Crawler/Stock data API	Monthly	Data Collection Data Preprocessing	Crawler Stock database Data analysis
	Shareholder data	Shareholder data crawler	Weekly		
	Technical data	Stock data API	Daily		
ML Models	Fundamental ML Model	ML training process	Monthly	Feature engineering Model training Model evaluation	Model training Stock recommend module
	Chip ML model	ML training process	Weekly		Stock filtering module
	Technical ML model	ML training process	Daily		Stock trading point recommendation module

The ML process and the software development process are not a waterfall model. Their workflows are iterative. There are many feedback loops in their workflow. For example, the step of model training may loop back to feature engineering, the model evaluation and deployment may loop back to any of the previous ML stages, and the detail design may loop back to architecture design. Those loop back cycles make the process of the machine learning software engineering more complex but we can get better understanding by clarify the sequence of ML stages and software development stages.

The relationship between ML processes and software development processes is shown in Table 5. The x-axis is the software development process and the y-axis is the ML process. The sign –1 and 1 mean one process before or after the other process. For example, the relationship between "Model requirement" and "Architecture design" is –1. This indicate we have to do "Model requirement" first then the "Architecture design" can be done. In our implementation, we complete "Architecture design" in software process first, then the ML process "Feature engineering" start, so their relationship is "1". The symbol "0" means both processes is on the same time period. For example, the ML process "Data preprocessing" and the software development process "Detail design" can do on the same time period.

Table 5. The relationship between ML processes and software development processes.



	Requirement analysis	Architecture design	Detail design	Implementation and unit testing	Integration and system testing
Model requirements	0	-1	-1	-1	-1
Data collection	1	1/0	0	0	-1
Data preprocessing	1	1/0	0	0	-1
Feature engineering	1	1	0	0	-1
Model training	1	1	0	0	-1
Model evaluation	1	1	0	0	-1
Model deployment	1	1	1	0	-1

4.2 A Machine Learning Software Engineering for Stock Investment Software

Considering the analysis and design process of the intelligent stock investment system, a workflow diagram of machine learning software engineering is shown in Fig. 8. The stages on the left side are ML processes. The software development processes are displayed on right side of the diagram.

After requirements acquisition and basic requirements analysis done, analyzing the model requirements of the ML model can be started. These basic requirements analysis results have been shown in the use case diagram of Fig. 2. In our prototype implementation, the ML modules and software modules are implemented by different teams. For better work division and team cooperation we design the system architecture and module interface first. After the architecture design has been done, the ML team start to design programs for data collection, data preprocessing, model training and model evaluation. The software development team start to do the detail design on the same time.

The final product of ML team in this stage not only includes well-trained ML models but also includes the workflow about how to retrain those models for future ML models updating. The final product of the software development team includes the interface of ML models. For example, the StockSieveStrategy interface shown in Fig. 6 is related to the fundamental, chip analysis, and technical ML models. The steps of detail design, software implementation, and ML model training and evaluation are iterative. The ML team can deliver earlier version of ML models to software development team for system implementation and testing and deliver fine-turned ML models in the future. After ML models and software prototype has been done, the integration and system testing can be started. Similar with software module, the ML model might out of date because following the stock data renewed the old ML model might cannot predict new situation well. The model monitoring stage is used to represent such situation and the data analyzer or the system can update ML models in this moment.

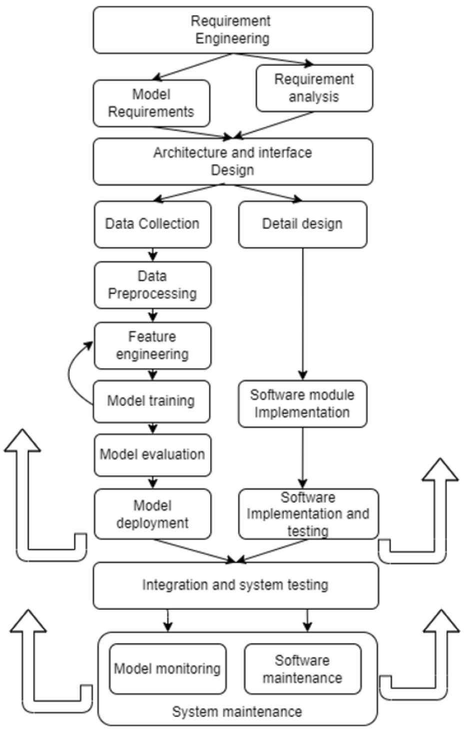


Fig. 10. A brief processes diagram of machine learning software engineering.

4.3 Stock Investment Software Design as a Utility Optimization Problem

There are more than 1000 stocks in Taiwan stock exchange market, using a ML model for all stock prediction is impossible and training models for every stock is unreasonable also. Basically, an investor only interested a few investing targets, not all stocks in the market. Considering this issue, we can use a “lazy” strategy to solve the problem. The system only train ML model assigned by the data analyzer in advance. When an investor wants to know suggestions of a new stock, then the system start to train ML models for the stock if the ML models do not exist or expired. The system only store *N* models. If the number of models bigger than *N*, one model will be deleted to maintain the storage space. The shortage of the strategy is longer response time if the stock is accessed in first time.

Although the lazy strategy is work, it is just a design strategy. Taking executing time and storage space into account, we formulate those consideration to a utility optimization problem.

Assume that a machine learning software *S* has machine learning models M_1, M_2, …, M_n. Each model can be either reserved or on-demand depended on software analysis. The on-demand model design logic is described briefly in the following statements:

- The amount of available space must be greater than the maximum of storage space of on-demands models.
- The server response time must be reasonable if users submit on-demand requests.

Some assumptions are made to formulate the model, including a fixed training time of machine learning model, a fixed storage size of machine learning mode and a fixed response time of the other service module ignoring network latency. Notations are defined as follows, and will be used in our research.

Notations

T_i	The training time of model *i*
S_i	The storage size of model *i*
δ_i	The indicator of whether model is on-demand or not
α	The trade-off coefficient between time and space
τ	The thread number of the deployed server
B	The available buffer space for on-demand model
S	The total storage size of the other service modules
ST	The total storage threshold of the deployed server
T	The computational time of the other service modules
RT	The service response time threshold

With these notations, the utility optimization problem can be defined as follows:

$$\min_{\delta}\left\{\alpha(\sum\delta_i T_i/\sum T_i)+(1-\alpha)(\sum(1-\delta_i)S_i/\sum S_i)\right\}.$$

Subject to
1. $\delta_i \in \{0, 1\}$,
2. $\alpha \in \{0, 1\}$,
3. $T_i > 0$,
4. $S_i > 0$,
5. $\tau \in \mathbb{N}$.
6. $\sum_i (1-\delta_i)S_i + B + S \leq ST$,
7. $B \geq \max_i\{\delta_i S_i\}$,
8. $\max_i\{\delta_i T_i\} + T \leq RT$
9. $\frac{1}{\tau}\sum\delta_i T_i + T \leq RT$

This utility optimization problem aims to minimize the trade-off between the training time of on-demand models and the storage size of reserved models. In order to simultaneously compare the effect of training time and storage size, the normalization technique is applied to scale both items. Constraint 1 is a binary constraint and states that machine learning model can be either reserved or on-demand. Constraint 2 states that the trade-off coefficient is between 0 and 1. Note that if $\alpha = 1$, it means that server storage is limited and all model are on-demand. On the other hand, if $\alpha = 0$, the deployed server storage is enough for all machine learning models. Constraint 3 and 4 ensure that the training time and the storage size should be positive. Constraint 5 ensures that the thread number of the deployed server should be a nature number. Constraint 6 and 7 state that the deployed server needs to have available space for at least one on-demand models. Constraint 8 states that the service response time of an on-demand request must be reasonable. Constraint 9 states that the service response of simultaneous on-demand requests must be reasonable.

The utility optimization problem is a binary optimization problem which is NP-hard [12-14]. Therefore, there is no polynomial-time solution. It is seldom to analyze all the stocks in the stock markets. There are many heuristic-based approaches for selecting potential targets and we can use those approach to decide which stock can be choose to create its ML model.

In our approach, we use stock chip analysis result and fundamental information to select candidate target and a lazy strategy to train ML model. Other approaches are brief describe as follows.

- Top-*k* market value: Many researches have demonstrated that the positive relationship between market value and return of rate [15, 16]. It is a common used approach in stock market. The famous Taiwan 50 ETF is based on the top-50 market value heuristics.
- Long/short average order status: Moving average (MA) has won its reputation in forming trading strategies [17, 18]. The relationships between different MAs can be proxy variables of price trends. For example, an average order status is long, if short-MA, medium-MA and long-MA are at the descending order. Long/short average order status are very strong signal for the uptrend/downtrend of stock prices.
- Top-*k* news: News sentiment is correlated to investors' sentiments and attentions, and hence affect stock prices. Numerous researches have studied sentiment-based trading strategies. [19, 20] In stock market, news sentiment can be used to identify the potential popular stocks.

5. CONCLUSIONS

This paper introduces the design and implementation of an intelligent system for stock investment decision making. A workflow of machine learning software engineering is discussed in the paper. We explore the relationship between ML process and software development process by analyzing the design and implementation process of ML related modules. Although different ML based software might have different development workflow, the experience proposed in this paper can contribute to the area of machine learning software engineering.

There are three important time points when developing a machine learning based software for the software development teams work division: requirement analysis, design, and maintenance. In requirement analysis stage, the ML team starts to analysis ML models' requirements and the software development team starts to analysis software requirements. This is the first work division of ML team and software development team. In the design stage, the ML team starts to collect and process data. The software development team can design the software modules on the same time. After the interface between ML model and software module has been designed, training and evaluating of ML models and software modules implementation and testing can be processed simultaneous. Because the ML models might out of date frequently, the ML models should be monitored and updated if it is not work well. Model monitoring and updating are important issues in the machine learning software engineering. The software development team might have to consider those issues in the requirement analysis stage and design related interface for model updating. The strategy design pattern is helpful to those requirements.

Executing time and storage space are two important issue when design a ML based software. In this paper we propose a utility optimization problem to provide the designer a way to consider trade-off between executing time and storage space.

There are still more challenges for the machine learning software engineering such as how to do the integrating and system testing in a smooth way between ML model development and software module development. Based on the results of this paper, we can explore those issues in the future.

REFERENCES

1. L. P. Marcos, *Advances in Financial Machine Learning*, Wiley, NY, 2018.
2. K. Lano and H. Haughton, *Financial Software Engineering*, Springer Cham, 2019.
3. S. Amershi *et al.*, “Software engineering for machine learning: A case study,” in *Proceedings of the 41st International Conference on Software Engineering in Practice*, 2019, pp. 291-300.
4. M. S. Rahman, E. Rivera, F. Khomh, Y.-G. Guéhéneuc, and B. Lehnert, “Machine learning software engineering in practice: An industrial case study,” *arXiv Preprint*, 2019, arXiv:1906.07154.
5. A. Serban, K. van der Blom, H. Hoos, and J. Visser, “Adoption and effects of software engineering best practices in machine learning,” in *Proceedings of ACM International Symposium on Empirical Software Engineering and Measurement*, 2020, pp. 1-12.
6. H. Washizaki, *et al.*, “Software-engineering design patterns for machine learning applications,” *Computer*, Vol. 55, 2022, pp. 30-39.
7. G. L. Morris, *Candlestick Charting Explained: Timeless Techniques for Trading Stocks and Futures*, 2nd ed., McGraw-Hill Trade, NY, 1995.
8. T. Y. Yu, “The relationship between change in the number of shareholders and stock return”, Master Thesis, Department of Finance, National Chengchi University, 2017.
9. https://www.tdcc.com.tw/smWeb/QryStockAjax.do
10. XGBoost https://arxiv.org/abs/1603.02754
11. M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley Professional, NY, 2002.
12. R. S. Pavithr and Gursaran, “Quantum inspired social evolution (QSE) algorithm for 0-1 knapsack problem,” *Swarm and Evolutionary Computation*, Vol. 29, 2016, pp. 33-46.
13. Z. Yang, B. W.-K. Ling, and C. Bingham, “Extracting underlying trend and predicting power usage via joint SSA and sparse binary programming,” in *Proceedings of IEEE International Symposium on Circuits and Systems*, 2013, pp. 1312-1315.
14. Md. A. K. Azad, A. M. A. C. Rocha, and E. M. G. P. Fernandes, “Improved binary artificial fish swarm algorithm for the 0-1 multidimensional knapsack problems,” *Swarm and Evolutionary Computation*, Vol. 14, 2014, pp. 66-75.
15. R. W. Banz, “The relationship between return and market value of common stocks,” *Journal of Financial Economics*, Vol. 9, 1981, pp. 3-18.
16. “Analysis of the effect of economic value added and market value added on stock

return on the Indonesian stock exchange case study on health companies in 2019-2020," *Hasanuddin Journal of Applied Business and Entrepreneurship*, http://journal.unhas.ac.id/index.php/hjabe/article/view/21662, 2022.
17. E. Pätäri and M. Vilska, "Performance of moving average trading strategies over varying stock market conditions: the Finnish evidence," *Applied Economics*, Vol. 46, 2014, pp. 2851-2872.
18. J.-Z. Huang and Z. (James) Huang, "Testing moving average trading strategies on ETFs," *Journal of Empirical Finance*, Vol. 57, 2020, pp. 16-32.
19. D. E. Allen, M. McAleer, and A. K. Singh, "Daily market news sentiment and stock prices," *Applied Economics*, Vol. 51, 2019, pp. 3212-3235.
20. W. Zhang and S. Skiena, "Trading strategies to exploit blog and news sentiment," in *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, 2010, pp. 375-378.

Chiung Hon Lee (李俊宏) received the Ph.D. degree in Electronic Engineering from the National Chung Cheng University in Taiwan in 2006. He is an Associate Professor at Department of Computer Science and Information Engineering, Feng Chia university in Taiwan. His research interests are in fuzzy time series, machine learning software engineering, and machine learning for investment decision making.

Tsung-Ju Lee (李宗儒) received the Ph.D. degree in Computer Science and Engineering from National Chiao Tung University, Taiwan, in 2013. He is an Assistant Professor in the Department of Finance at Feng Chia University. He is also the Deputy Director of FinTech Research Center at Feng Chia University. His research interests include FinTech, green finance, algorithmic trading, machine learning and data mining.

Yu-Sheng Chiang (姜育昇) is an MS student at Department of Computer Science and Information Engineering, Feng Chia University in Taiwan. He received his Bachelor degrees in the Department of Statistics from Feng Chia University, Taiwan. His research interests include machine learning, investment decision making, and statistical test applications.

Shih-Yi Yuan (袁世一) received his Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 1997. He joined the Communications Engineering Department of Feng Chia University, Taichung, Taiwan in 2010 as an Associate Professor. He is also a member of ICEMC center of Feng Chia University. His research interests include driver design, algorithm design for digital system EMI estimation, and software IC-EMC model development for PI, EMI, or EMS of a digital system.

Jyh-Hwa Liou (劉智華) is an Assistant Professor at the Department of Finance, Feng Chia University in Taiwan. She received her Ph.D. degree from the Institute of Information Management, National Chiao Tung University in Taiwan. Her research interests include machine learning, deep learning, social computing and financial technology.