Module Interface Specification for Stock Prediction System

Renjie Zhang

December 7, 2017

1 Revision History

Date	Version	Notes
22/11/2017	1.0	Create
28/11/2017	1.1	Update

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/renjiezhang/CAS-741/blob/master/Doc/SRS/SRS.pdf

[Do you actually use any of your SRS symbols? I had a quick look and I didn't see any examples. —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for Stock Prediction System which is used to predict the future stock price based on the historical data. Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/renjiezhang/CAS-741.

4 Notation

The structure of the MIS for modules comes from **?**, with the addition that template modules have been adapted from **?**. The mathematical notation comes from Chapter 3 of **?**. For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Stock Prediction System.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
Python List	list	a python list consists of numbers and string
Python Plot	plot	a GUI plot generated by the system

[It is better to not specify a specific programming language, like Python. Stay language neutral as long as you can. Also, Python lists have all elements of the same type, and a plot is not a type. —SS]

The specification of Stock Prediction System uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Stock Prediction System uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

M1: Hardware-Hiding Module

M2: Main Module

M3: Data Input Module

M4: Price Volatility Module

M5: Price Momentum Module

M6: Kernelling And Prediction Module

M7: Plot Module

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding Module	Main Module Data Input Module Price Volatility Module Price Momentum Module Kernelling And Prediction Module
Software Decision Module	Data Plot Module

Table 1: Module Hierarchy

[Kernelling isn't a software decision module? —SS]

6 MIS of Main Module (M2)

6.1 Module

Main

6.2 Uses

Hardware-Hiding Module (M1)

6.3 Syntax

NA

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
main	-	$\mathbb{R}$	-

6.4 Semantics

6.4.1 State Variables

NA

6.4.2 Access Routine Semantics

main():

- transition: Calls the Data Input Modules, Volatility Module, Momentum Module and Output Module with parameters and retrieve the returned data from them.
- output: The percentage of the prediction result. The range of the score is between -1 to 1. if the scoring is negative then the price is going to decrease otherwise increase.
- exception: NA

[There is not enough information here to build main. Usually a module will only have a transition or an output. Do you really mean output? This implies that your program will return a single real number to whatever program calls main. For your transition you should show the access program calls that will be used to modify the state of the other modules. —SS]

[It is nice if each new module starts on a new page. —SS]

7 MIS of Data Input Module (M3)

7.1 Module

Input Module

7.2 Uses

Main Module (M2)

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
ReadCSV	char	list	IOError

[The type char is a single character. You probably mean string. —SS]
[The output type list is ambiguous. What is the type of the elements in the list? —SS]
[If you are reading from a file, you need an environment variable to represent the external file. —SS]

7.4 Semantics

7.4.1 State Variables

NA

7.4.2 Access Routine Semantics

ReadCSV():

- transition: Load the CSV file by the input file name and read the data of the date and close price column. [This is not a transition. A transition changes the state of some module, or an environment variable. —SS]
- output: A python array of the date and price [Do you need to define a new record type that stores date and price? You could then have a list of elements of this new type. —SS]
- exception: Invalid file name and path. Invalid column name and data format in the file. [What is the name of the exception? It should be shown here as well. —SS]

8 MIS of Price Volatility Module (M4)

8.1 Module

Volatility Module

8.2 Uses

Data Input Module (M3)
Main Module (M2)

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
GetPriceVolatility	$\mathbb{R}^n$,$\mathbb{N}^n$	$\mathbb{R}^n$	NotFittedError

8.4 Semantics

8.4.1 State Variables

NumberOfDays: Number of Days

Prices: A list of real numbers. It is the stock price of each days.

PriceVolatility : A list of real numbers which represents the price valitility

8.4.2 Access Routine Semantics

GetPriceVolatility():

- transition: Calculate the price volatility based on the input parameters, number of days and prices.
- output: A python array of the price volatility
- exception: Improper price such as negative number.

9 MIS of Price Momentum Module (M5)

9.1 Module

Momentum Module

9.2 Uses

Data Input Module (M3)
Main Module (M2)

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
GetPriceMomentum	$\mathbb{R}^n$,$\mathbb{N}^n$	$\mathbb{R}^n$	NotFittedError

9.4 Semantics

9.4.1 State Variables

NumberOfDays: Number of Days

Prices: A list of real numbers. It is the stock price of each days.

PriceMomentum : A list of real numbers which represents the price momentum

9.4.2 Access Routine Semantics

GetPriceMomentum():

- transition: Calculate the price momentum based on the input parameters, number of days and prices.
- output: A python array of the price momentum
- exception: Improper price such as negative number.

10 MIS of Kernelling And Prediction Module (M6)

10.1 Module

Predict Module

10.2 Uses

Data Input Module (M3)
Main Module (M2)
Volatility Module(M4)
Momentum Module(M5)

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
Predict	char, $\mathbb{R}^n$,$\mathbb{N}^n$	$\mathbb{R}$	-
SVC	char	-	-

10.4 Semantics

10.4.1 State Variables

Predict()
Code: A string type variable represents the permanent code of the company
NumberofDays: The number of days from the records

PriceVolatility: The price volatility list calculated from the Price Volatility Model

PriceMomentum: The price momentum list calculated from the Price Momentum Model

Score : The percentage of the prediction result. The range of the score is between -1 to 1. if the scoring is negative then the price is going to decrease otherwise increase.

SVC()
Kernel mode: kernel='rbf

10.4.2 Access Routine Semantics

Predict():

- transition: Calculate the array of the SVM for each company in the file using the result of price volatility and price momentum.
- output: A python array of the score
- exception: NA

11 MIS of Plot Module (M7)

11.1 Module

Plot Module

11.2 Uses

Data Input Module (M3)
Main Module (M2)

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
Plot	char, $\mathbb{R}^n$	plot	-

11.4 Semantics

11.4.1 State Variables

Dates: The array of the dates of each record

Prices: The array of the prices of each record

11.4.2 Access Routine Semantics

Plot():

- transition: Plot the graphic interface of the record.
- output: The plot with date on the x ray and price on the y ray.
- exception: NA

11.5 Semantics

11.5.1 State Variables

Dates: The array of the dates of each record

Prices: The array of the prices of each record

11.5.2 Access Routine Semantics

Plot():

- transition: Plot the graphic interface of the record.

- output: The plot with date on the x ray and price on the y ray.
- exception: NA

[There is not enough detail in your module specifications. No programming could be done with the starting point of your MIS. The equations that are in your SRS should appear in the MIS as well. Your modules should either have a transition or an output, not both. Your type information isn't consistent and is often missing. You need environment variables for files and for the screen. The screen environment variable is used when you generate a plot. If you have state variables, there should be a transition specified that changes the state of those state variables. There also needs to be an obvious way to set their initial state. —SS]

References

NA [You have two citations that you didn't include in your bib file. —SS]

12 Appendix

NA [You could remove this section if you don't use it. —SS]