Stock Predict System

Renjie Zhang

December 17, 2017

1 Revision History

Date	Version	Notes
17/12/2017	1.0	Create

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
SVM	Support Vector Machine
RDD	Resilient Distributed Datasets
K	Kernel function
X	features of the data (date , price)
C	The price from different days
β	The error modifier
y	The result between 1 and -1

Contents

List of Tables

List of Figures

This document provides a testing report of Stock Prediction System. This document covers both system testing and unit testing. Traceability between testing and both requirements and modules is given in the final section. The implementation of the tests in this report follows the Test Plan document. For more information please view my documentation at the repository: https://github.com/renjiezhang/CAS-741

3 Functional Requirements Evaluation

3.1 Data Input

The part of test is to verify the loading a dataset from a CSV file. This test ensures the data input method and the input data.

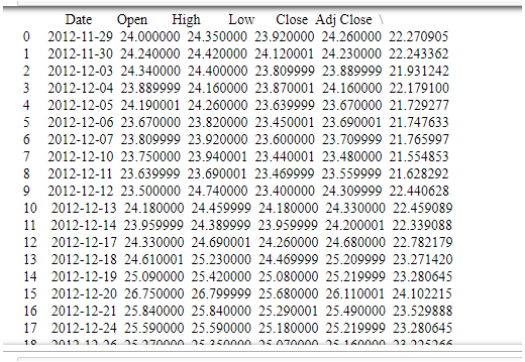
File loading Testing

1. File is loaded successfully

   Type: Functional, Manual, Static
   Initial State: NA
   Input: File Path
   Output: The content of the file
   How test will be performed: System tries to load the data set file based on the file name and location without issues.
   Result : Pass. The screen shot shows the result of the data input.

2. Input Data Validation
   Type: Functional, Manual
   Initial State: NA
   Input: Data from the file
   Output: The content of the file.
   How test will be performed: System have to ensure the data type and format is correct for each columns of the file : the pattern of the date, the formats of the price and number of digits of decimals. If the format does match the requirement, the program will encounter an IO exception. The example of the CSV file is shown in the following figure.

Result: Pass. The screen shot shows the result of the data input. The function reads the CSV file with proper format.



	Date	Open	High	Low	Close	Adj Close
0	2012-11-29	24.000000	24.350000	23.920000	24.260000	22.270905
1	2012-11-30	24.240000	24.420000	24.120001	24.230000	22.243362
2	2012-12-03	24.340000	24.400000	23.809999	23.889999	21.931242
3	2012-12-04	23.889999	24.160000	23.870001	24.160000	22.179100
4	2012-12-05	24.190001	24.260000	23.639999	23.670000	21.729277
5	2012-12-06	23.670000	23.820000	23.450001	23.690001	21.747633
6	2012-12-07	23.809999	23.920000	23.600000	23.709999	21.765997
7	2012-12-10	23.750000	23.940001	23.440001	23.480000	21.554853
8	2012-12-11	23.639999	23.690001	23.469999	23.559999	21.628292
9	2012-12-12	23.500000	24.740000	23.400000	24.309999	22.440628
10	2012-12-13	24.180000	24.459999	24.180000	24.330000	22.459089
11	2012-12-14	23.959999	24.389999	23.959999	24.200001	22.339088
12	2012-12-17	24.330000	24.690001	24.260000	24.680000	22.782179
13	2012-12-18	24.610001	25.230000	24.469999	25.209999	23.271420
14	2012-12-19	25.090000	25.420000	25.080000	25.219999	23.280645
15	2012-12-20	26.750000	26.799999	25.680000	26.110001	24.102215
16	2012-12-21	25.840000	25.840000	25.290001	25.490000	23.529888
17	2012-12-24	25.590000	25.590000	25.180000	25.219999	23.280645

Figure 1:

3.2 Plot Testing

Type: Functional, Manual, Static Initial State: NA
Input: A dataset file
Output: A correct plot
How test will be performed: System reads the data and generate a plot using the price and the date as the x and y axis.
result : Pass. The screen shot shows the result of the plot. It displays the historical data of four companies from 2013 to the end of 2017.

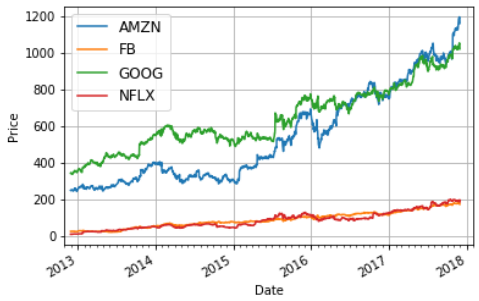


Figure 2:

3.3 Distributed System

This section focus on the distributed system of Spark platform. It guides the users to test the data flow on the spark system. RDD is the format of data flow in Spark. testers need to double check the data was distributed into each workers through RDD.

Spark RDD Testing

Type: Functional, Manual, Static
Initial State: NA
Input: There will be a list of numbers to input to the RDD function. To test the RDD function, the driver will ask Spark to return some random number back.
Output: The list of numbers which is collected back from workers.
How test will be performed: There will be an input array to the driver, driver assign the array to each workers. Each worker receives part of the numbers in the list. When the driver collect the RDD, works return each part of the number back to driver.
Result: Pass. The screen shot shows the system assign an array of integers from 0 to 1000 and collect 5 samples back.

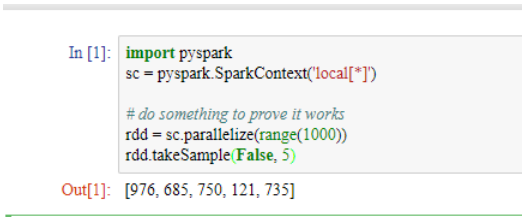


Figure 3:

4 Nonfunctional Requirements Evaluation

4.1 Usability

This test needs to be done under a Spark environment. Docker container is suggested. Python and Spark are required to be installed in the container and the following frameworks are required for Python :
pandas, numpy, sklearn, matplotlib and pyspark

1. Participants
   Classmate: Li Shusheng

2. Document given
   https://github.com/renjiezhang/CAS-741/blob/master/src/InstallationInstructic
   pdf

3. Task for participants
   Read the Installation Instruction, install the necessary tools and frameworks and run the test successfully.

4. How test will be measured and performed
   If all participants can run test case successfully with the instruction, then Usability of the program is good.

5. Result
   Docker for windows does not work on windows home edition, it works on professional edition since it needs the Hyper-V feature to be turned on. Everything else works fine with a proper windows edition.

4.2 Performance

Type: Manual
Initial State: NA
Input/Condition: NA
Output/Result: Result of the time cost
How test will be performed:
Chang the size of the dataset files by downloading the files with different date range. For example, reduce the date range from 5 years to 3 years and compare the performance.
Change the number of date parameter to predict short term and long term stock price. By adding a new type of number of date, the software need to increase the run time.
Result: The prediction accuracy was influenced by the days of the prediction. The longer term (90,270) is more accurate then the shorter term(1,5).
The performance of 5 years dataset file with 4 types of prediction dates is about 36 seconds. By reducing the prediction date types from 4 to 3, the execution time become to 21 seconds.
The performance result is on the result.txt

5 Comparison to Existing Implementation

The test on multiple spark workers are removed due to the limit time because I have encountered some problems with configuration of multiple Docker containers. The system for now is on a single node spark machine.

6 Unit Testing

The unit testing plan will involve the following modules: Load Data, SVM Kernelling, Volatility Calculating, Momentum Calculating, Predict and Output.

1. Calculate Volatility
   This function is use to calculate the price volatility and index volatility. Since the calculation is similar they share the same function. A correct result is expected by inputting the parameters to the equation

$$\frac{\sum_{i=t-n+1}^{t} \frac{C_i - C_{i-1}}{Ci-1}}{n}$$

Input: A pre-defined Array of stock record, which contains two elements: price and date
Output: An array of price volatility based on the input array

2. Calculate Momentum
This function is use to calculate the price momentum and index momentum. Since the calculation is similar they share the same function. A correct result is expected by inputting the parameters to the equation
$$\frac{\sum_{i=t-n+1}^{t} \frac{C_i - C_{i-1}}{Ci-1}}{n}$$

Input: A pre-defined Array of stock record, which contains two elements: price and date
Output: An array of price volatility based on the input array

3. Predict
The Predict module receives a set of parameters calculated from the previous functions and returns a result. A correct result is expected by inputting the parameters to the equation

$y = \beta_0 + \sum a_i y_i K(x(i), x)$
Input: Two pre-defined lists of stock record, which contains two elements: price and date. They have same format, one is for stock price and another is for NASDQ Index record
Output:A score of a decimal number which represents the probability

The result of the unit test is shown as the screen shot:

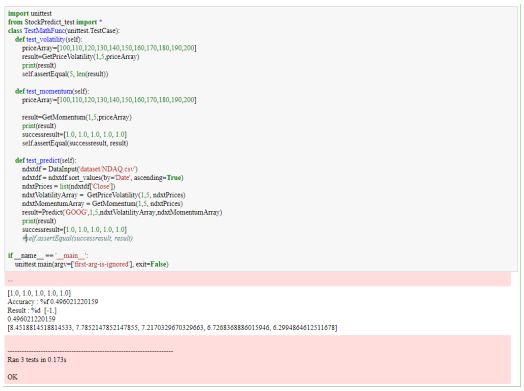


Figure 4:

7 Changes Due to Testing

Change some variables from hard coding values to constant
Change function calls to be more flexible

8 Automated Testing

NA

9 Trace to Requirements

R1: Input data

R2: Polt

R3: SVM Calculation

R4: Verification

R5: Output

	R1	R2	R3	R4	R5
Input Test	x	x		x	
Plot Test		x			
Spark test					
Volatility Test		x			x
Momentum Test		x			x
Predict Test		x			x

10 Trace to Modules

M1: Hardware-Hiding Module

M2: Data Input Module

M3: Kernelling Module

M4: Price Volatility Module

M5: Price Momentum Module

M6: Stock Prediction Module

M7: Output Module

M8: Spark RDD Module

M9: Data Plot Module

	M1	M2	M3	M4	M5	M6	M7	M8	M9
Input Test		x							x
Plot Test									x
Spark test								x	
Volatility Test				x					
Momentum Test					x				
Predict Test				x	x	x	x		

11 Code Coverage Metrics

NA