Stock Prediction System

Renjie Zhang

December 17, 2017

1 Revision History

Date	Version	Notes
2017-10-06	1.0	create
2017-10-09	1.1	update

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

3 General Information

This Document is a test plan of the software-Stock Prediction System which is a tool used to predict the stock prices using machine learning. The software will be worked on a distributed system.

3.1 Purpose

This test plan describes the testing methods that will drive the testing of the Stock Prediction System and give a guide to the users about the QA. This document includes the descriptions of testing tool, testing functions, unit testing method and system test. Based on these testing, the users may cache the functional and non functional issues from the first release of the software and find out the improvement.

3.2 Scope

The testing plan will cover both system testing and unit testing. This software is wrote by Python. The functions are the data input, data format validation, calculation of SVM, data plot, Spark RDD Distributed System, and the output of the result. Basically every part of the system will be tested, and most of the testing will be done by unit testing. For more information please read the SRS in my repository: `https://github.com/renjiezhang/CAS-741`.

3.3 Overview of Document

This document explains the testing plan for the software - Stock Prediction System. It includes the briefcase of software description, the different testing methods - unit testing and integration testing, testing tools. It will cover both functional and non functional requirements.

4 Plan

In this section, the general description of the testing plan for the software and corresponding testing tools will be introduced.

4.1 Software Description

The Stock Prediction System is used to analyze the future trend of stocks. The prediction was provided by machine learning algorithms based on the historical data. The system will be run on a big data platform (Spark), to obtain the more accurate results. In this case, we need to setup a distributed system to support Spark.

4.2 Test Team

Renjie Zhang

4.3 Automated Testing Approach

NA

4.4 Verification Tools

Python unit testing : This framework was included in Python standard library. It is easy to use by people familiar with the xUnit frameworks, strong support for test organization and reuse via test suites
Wing : A popular Python IDE which contents the code checking and indent correction.

4.5 Non-Testing Based Verification

NA

5 System Test Description

This section describes the testing on the system environment, such as the data file input and spark distribution system. Users need to make sure the application run on the system without any issues.

5.1 Tests for Functional Requirements

The functional requirements includes the data input, the data format verification, the Spark RDD transaction, the plot generation and the display of the output. The software must fulfill all of the functional requirements without issues.

5.1.1 Data Input

The system needs to load a dataset from a CSV file. This test ensures the data input method.

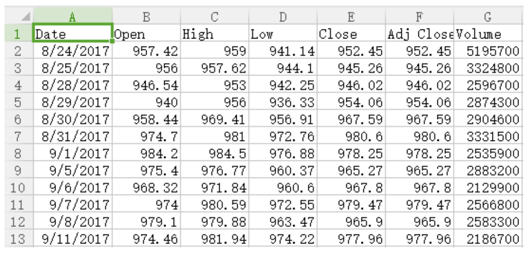
File loading Testing

1. File is loaded successfully

   Type: Functional, Manual, Static etc. Initial State: NA Input: File Path Output: Successful Message How test will be performed: System tries to load the data set file based on the file name and location without issues.

2. Input Data Validation

   Type: Functional, Manual
   Initial State: NA
   Input: Data from the file
   Output: A successful message.
   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.



	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	8/24/2017	957.42	959	941.14	952.45	952.45	5195700
3	8/25/2017	956	957.62	944.1	945.26	945.26	3324800
4	8/28/2017	946.54	953	942.25	946.02	946.02	2596700
5	8/29/2017	940	956	936.33	954.06	954.06	2874300
6	8/30/2017	958.44	969.41	956.91	967.59	967.59	2904600
7	8/31/2017	974.7	981	972.76	980.6	980.6	3331500
8	9/1/2017	984.2	984.5	976.88	978.25	978.25	2535900
9	9/5/2017	975.4	976.77	960.37	965.27	965.27	2883200
10	9/6/2017	968.32	971.84	960.6	967.8	967.8	2129900
11	9/7/2017	974	980.59	972.55	979.47	979.47	2566800
12	9/8/2017	979.1	979.88	963.47	965.9	965.9	2583300
13	9/11/2017	974.46	981.94	974.22	977.96	977.96	2186700

Figure 1:

3. Plot Testing

   Type: Functional, Manual, Static Initial State: NA
   Input: A pre-defined Array
   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.

5.1.2 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

1. Data is distributed to workers

   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. A sample work flow is shown below:

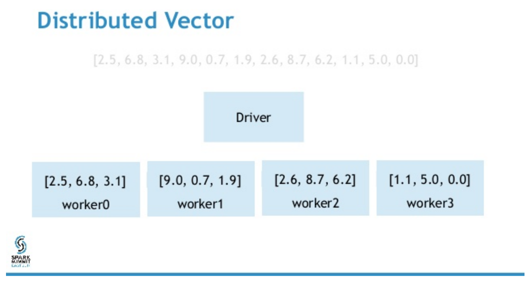


Figure 2:

5.2 Tests for Non functional Requirements

Non functional requirements are not as prior as functional requirements, but it is still necessary to have tests on them in order to improve the software performance and quality.

5.2.1 Big Data System Testing

In this section, the only requirement is the performance.

Performance Testing

1. Performance of the workers

   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.

5.3 Traceability Between Test Cases and Requirements

	Input	Plot	Data Validation	Calculation	Output	Performance
Input Data	X					
Data Validation	X		X			X
RDD	X					
Plot		X				X
Kernelling				X		X
Volatility				X		
Momentum				X		
Prediction				X	X	

Table 1: Traceability Matrix Showing the Connections Between Requirements and Test Cases

6 Unit Testing Plan

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 mo-

mentum. 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 inputing 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

7 Appendix

NA

7.1 Symbolic Parameters

NA

7.2 Usability Survey Questions?

NA