Stock Prediction System

Renjie Zhang

October 20, 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

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

Contents

List of Tables

List of Figures

This document ...

3 General Information

3.1 Purpose

This test plan describes the testing methods that will drive the testing of the Stock Prediction System. The document introduces:
Test Strategy: rules the test will be based on, including the givens of the project (e.g.: start / end dates, objectives, assumptions); description of the process to set up a valid test (e.g.: entry / exit criteria, creation of test cases, specific tasks to perform, scheduling, data strategy). Execution Strategy: describes how the test will be performed and process to identify and report defects, and to fix and implement fixes. Test Management: process to handle the logistics of the test and all the events that come up during execution (e.g.: communications, escalation procedures, risk and mitigation, team roster)

3.2 Scope

The testing plan will cover both system testing and unit testing. 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.

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

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), in order 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.
PyChecker (Optional) : A source checking tools which finds problems that are typically caught by a compiler for less dynamic languages; imports each module before checking it.

[Thoughts on what tools to use, such as the following: unit testing framework, valgrind, static analyzer, make, continuous integration, test coverage tool, etc. —SS]

4.5 Non-Testing Based Verification

NA

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Data Input

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: Successful message How test will be performed: System have to ensure the data type and format is correct for each column of the file such as the pattern of the date, the formats of the price and number of digits of decimals. Shown in Figure 1

	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:

5.1.2 Distributed System

Spark RDD Testing

1. Data is distributed to workers

   Type: Functional, Manual, Static etc. Initial State: NA Input: An Array from the driver Output: An updated array with marks of each worker How test will be performed: There will be an input array to the driver, driver assign the array to each workers. Workers add a mark with its identification to elements of the array and return it back to driver. Shown in Figure 2.

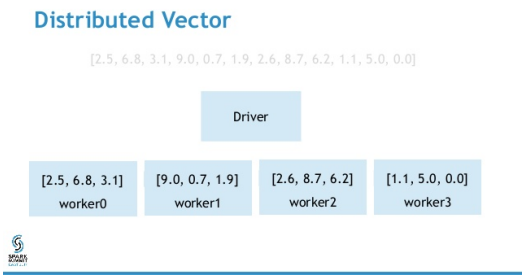

Figure 2:

5.1.3 Testing on The Algorithm

SVM Testing

1. Plot Testing

   Type: Functional, Manual, Static etc. 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.

2. SVM Calculation

   Type: Functional, Manual, Static etc. Initial State: NA Input: A collection of data Output: correct results How test will be performed:

The detail of the functions related to SVM Calculation can be found in Unit Test

5.2 Tests for Nonfunctional Requirements

5.2.1 Big Data System Testing

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: It is a good way to increase the performance by increase the number of works. Try to increase one more worker and compare the time cost on data training and testing.

5.3 Traceability Between Test Cases and Requirements

	Input Data	Plot	Data Validation	Calculation	Output	Performanc
Input Data testing	X					
Data Validation	X		X			
RDD Testing	X					
Polt testing		X				X
Kernelling				X		
Volatility Testing				X		
Momentum Testing				X		
Prediction Testing				X	X	
Performance Testing						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. Load Data
   The detail of the data loading was explained on 5.1.1

2. Kernelling
   In machine learning, kernel methods are a class of algorithms for pattern analysis, whose best known member is the support vector machine (SVM) RBF Kernel is used for price forecasting. A correct result is expected by inputing the parameters to the equation
   $k\left(X_i, X_k\right) = \exp\left(-\frac{1}{\delta^2}\sum_{n}^{j=1}(X_{ij} - X_{kj})^2\right)$

3. 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
   $\frac{\sum_{i=t-n+1}^{t} \frac{C_i - C_{i-1}}{Ci-1}}{n}$

4. 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
   $\frac{\sum_{i=t-n+1}^{t} \frac{C_i - C_{i-1}}{Ci-1}}{n}$

5. 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

$$y = \beta_0 + \sum a_i y_i K(x(i), x)$$

[Unit testing plans for internal functions and, if appropriate, output files —SS]

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.