# Test Report: Project Title

Author Name

December 17, 2017

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| Date 1 | 1.0 | Notes |
| Date 2 | 1.1 | Notes |

# 2 Symbols, Abbreviations and Acronyms

| symbol | description |
|---|---|
| T | Test |

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

# Contents


**1 Revision History** **i**

**2 Symbols, Abbreviations and Acronyms** **ii**

**3 Functional Requirements Evaluation** **1**

3.1 Data Input . . . . . 1

3.1.1 Distributed System . . . . . 2

**4 Nonfunctional Requirements Evaluation** **2**

4.1 Usability . . . . . 2

4.2 Performance . . . . . 2

4.3 etc. . . . . 2

**5 Comparison to Existing Implementation** **2**

**6 Unit Testing** **3**

**7 Changes Due to Testing** **4**

**8 Automated Testing** **4**

**9 Trace to Requirements** **4**

**10 Trace to Modules** **4**

**11 Code Coverage Metrics** **4**


# 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: Successful Message
   How test will be performed: System tries to load the data set file based on the file name and location without issues.
   Result : Pass

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.
Result: Pass

#### 3.1.1 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. A sample work flow is shown below:

# 4 Nonfunctional Requirements Evaluation

## 4.1 Usability

## 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.

### 4.3 etc.

## 5 Comparison to Existing Implementation

NA

## 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 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

4. 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.
   result : Pass

# 7 Changes Due to Testing

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.

8 Automated Testing

9 Trace to Requirements

10 Trace to Modules

11 Code Coverage Metrics