基于大数据的股票预测系统

摘要

本系统主要目的为通过计算机数据统计的方式对中国股票市场的单支股票进行预测，并寻找规律。使用多个接口提供的真实股票数据，进行分析演算，通过统计学以及神经网络的方式，找出股票的统计学与数学规律。其中主要使用的公式为**sigmoid函数，该公式主要用于神经网络的建模运算，对于本系统也起到了十分重要的作用。**目标则是超过80%以上的预测成功率，而本论文中得出的最终结果为一张图标，其中涨跌股票曲线体现出了十分明显的统计学区别。以下内容中包含了设计以及主要核心逻辑与技术的简要说明。有以上结果表明，股票预测存在可能性，并且涨跌股票拥有分别独立的统计学规律，通过观察图标，我们可以得出相应的结论。

Big Data Based Stock Forecasting System

ABSTRACT

The main purpose of this system is to predict single stocks in the Chinese stock market through computer data statistics and to find out the laws. The real stock data provided by multiple interfaces is used for analysis and calculation. Through statistics and neural networks, the statistical and mathematical rules of the stock are found out. The main formula used is the sigmoid function. This formula is mainly used in the modeling and calculation of neural networks and plays a very important role in this system. The goal is more than 80% of the predicted success rate, and the final result in this paper is an icon, which shows a very obvious statistical difference. The following content contains a brief description of the design and the main core logic and technology. The above results indicate that there is a possibility of stock prediction, and that stocks have separate statistical laws. By observing the icons, we can draw conclusions accordingly.

目录

[1 需求分析说明书 5](#_Toc515378456)

[1.1 引言 5](#_Toc515378457)

[1.1.1编写目的 5](#_Toc515378458)

[1.1.2背景介绍 5](#_Toc515378459)

[1.1.3定义 5](#_Toc515378460)

[1.2平台需求分析 6](#_Toc515378461)

[1.2.1社会需求 6](#_Toc515378462)

[1.2.2系统目标与愿景 7](#_Toc515378463)

[1.2.3系统面向用户类别和特性 8](#_Toc515378464)

[1.2.4运行环境 8](#_Toc515378465)

[1.2.5外部接口需求 9](#_Toc515378466)

[1.2.6功能需求 9](#_Toc515378467)

[1.2.7非功能性需求 11](#_Toc515378468)

[1.2.8软件需求规格表 12](#_Toc515378469)

[2 概要设计说明书 13](#_Toc515378470)

[2.1引言 13](#_Toc515378471)

[2.2 总体设计 14](#_Toc515378472)

[2.2.1需求 14](#_Toc515378473)

[2.2.2目标 14](#_Toc515378474)

[2.2.3运行环境 14](#_Toc515378475)

[2.2.4局限 14](#_Toc515378476)

[2.2.5结构 14](#_Toc515378477)

[2.3.接口设计 14](#_Toc515378478)

[2.3.1对外接口 14](#_Toc515378479)

[2.3.2内部接口 20](#_Toc515378480)

[2.4 功能模块设计 20](#_Toc515378481)

[2.4.1爬虫模块 20](#_Toc515378482)

[2.4.2基础数据服务模块 21](#_Toc515378483)

[2.4.3数据预测计算模块 22](#_Toc515378484)

[2.5 数据库设计 22](#_Toc515378485)

[2.5.1数据库结构设计 22](#_Toc515378486)

[2.5.2数据库环境 27](#_Toc515378487)

[2.6 程序模型设计 28](#_Toc515378488)

[2.6.1程序主流程 28](#_Toc515378489)

[2.6.2 运行预测与安全 28](#_Toc515378490)

[2.6.3安全与备灾 29](#_Toc515378491)

[3系统详细设计 30](#_Toc515378492)

[3.1 引言 30](#_Toc515378493)

[3.2 定义 30](#_Toc515378494)

[3.3整体框架与规范 31](#_Toc515378495)

[3.3.1整体框架 31](#_Toc515378496)

[3.3.2规范 31](#_Toc515378497)

[3.4 程序描述 32](#_Toc515378498)

[3.4.1总体包图 32](#_Toc515378499)

[3.4.2爬虫模块 33](#_Toc515378500)

[3.4.3计算模块 36](#_Toc515378501)

[3.4.4计算预测模块 40](#_Toc515378502)

[3.5限制与部署 41](#_Toc515378503)

[4 主要技术与关键模块实现 41](#_Toc515378504)

[4.1主要技术 42](#_Toc515378505)

[4.2 核心代码与算法 46](#_Toc515378506)

[4.2.1核心代码理论基础 46](#_Toc515378507)

[5 系统部署与结果 51](#_Toc515378508)

[5.1系统部署 51](#_Toc515378509)

[5.2结果 51](#_Toc515378510)

[6.总结部分 53](#_Toc515378511)

[致谢 54](#_Toc515378512)

[参考文献 55](#_Toc515378513)

# 1 需求分析说明书

## 1.1 引言

### 1.1.1编写目的

本文将提供给开发者最为基本的需求概述，希望通过本说明书，读者能够大致的整一个系统最为根本的需求，以及对于本系统有一个初步的了解，希望在阅读了本篇文章后，能够回答读者这些问题：这个系统是做什么的，这个系统想要的结果是什么样的以及这个系统需要什么样的环境运行等等。

### 1.1.2背景介绍

如今社会十分注重投资，股票投资作为一种经典并且广泛的投资方式被大部分人所接受，其中一部分的股民希望通过更加规律性的事物对于股票的涨跌进行预判，并得到能够长期稳定盈利的方式，而本系统则是为该部分人群制作，希望能够通过本系统的提供的数据，对大部分股票进行正确的预判。另一功能则是为股民提供一些信息的整合与汇总，本功能为次要功能。

|  |  |
| --- | --- |
| 系统名称 | 股票预测系统 |
| 系统提出者 | 徐智杰 |
| 开发者 | 徐智杰 |
| 主要用户 | 徐智杰 |

### 1.1.3定义

Spring Boot：一个可以轻松创建独立的，生产级别的给予Spring的应用程序，并且得到可以直接运行的.jar文件，同时对于第三方库也可以良好的适配。

MongoDB: 一种崭新的NoSQL数据库，可以灵活，贪心地存储实体与数据类型。

Ubuntu：一种[自由开源](https://en.wikipedia.org/wiki/Free_and_open-source_software)[的操作系统](https://en.wikipedia.org/wiki/Operating_system)和[Linux分发](https://en.wikipedia.org/wiki/Linux_distribution)基于[Debian的](https://en.wikipedia.org/wiki/Debian)系统。

Java：Java的是一个通用的计算机编程语言，也就是并行，基于类的，面向对象的，而专门设计，具有尽可能少的实现依赖越好。它旨在让应用程序开发人员“ 一次编写，随处运行 ”

IDEA编译器：IntelliJ IDEA是用于开发计算机软件的Java 集成开发环境（IDE）。它由JetBrains（以前称为IntelliJ）开发。

## 1.2平台需求分析

### 1.2.1社会需求

随着商场经济的逐渐深入，现代人对于资产的投资理念提高到了一个新的高度，而股票作为一种重要的投资手段，被人们所广泛接受和使用，电视节目与网络节目也都有大量的以“通过股票的形式盈利”的内容，这也从侧面反映出了，股民对于股市的期待以及对于能够通过一些途径获得盈利信息的心态。

下图为“上海证券交易所”2017年至2012年概况数据（数据来源为上交所<http://www.sse.com.cn/market/stockdata/overview/yearly/>）

我们可以明显的看到股票市场处2015年牛市以外均保持一定幅度的上升趋势，股票的总价值上升，可以从侧面体现股票市场的盈利存在一定的规律。

而希望知道或找出这个规律的人也自然而然的应运而生，再结合时代互联网经济的特征，我们处理数据的手段也不断提升，使用科学技术对于现有数据进行分析并找出规律，这是一个十分通用的方式。这一样也是本系统被设计的初衷与想法。

### 1.2.2系统目标与愿景

本系统的主要目的是开发一款能够比较准确预计股票走势，并且提供大部分股票数据的后台项目。

对于用户，他们可以通过调用这个接口获取部分特定股票的历史数据，并且获得股票的预测信息，或者他们也可以对于系统提供的预测信息，加入自己的判断与想法。

根据本项目的计划，系统主要分成以下两部分

1. 数据爬取与展现
2. 数据分析与预测

主要目标有以下几点

1. 数据抓取：

股票历史信息

上市公司企业信息

股票指数信息

股票其他必要信息

2、数据展示

调用接口展示JSON格式数据

1. 数据分析

使用部分技术对于已获得的数据进行分析，分析预测股票走势（调用接口）

### 1.2.3系统面向用户类别和特性

系统主要面向涉及股票交易的个人投资者，也就是传统意义上的股民或称之为散户群体，倾力投资国内股票市场，希望通过股票交易些许赢利以养家糊口，但不得不或长或短地担任股东，在交易大厅中曾经汗流浃背，曾经瑟瑟发抖，被社会另眼视为赌徒，被大股东视为小猪，且屡受绝望煎熬，无力自救，只能寄希望于政府那样的一个弱势人群。

其特点主要为：具有[分散性](https://baike.baidu.com/item/%E5%88%86%E6%95%A3%E6%80%A7/7479876)和流动性的特点。并且不论男女老少，不分文化职业，只要有购买能力都可以参与。因为门槛低，也导致了大部分用户不具备专业的投资水平和更多的投资时间，这也是大部分股民赔钱的根本性原因。所以，要想赚钱必须：1、需要花费更多的时间和精力；2、提高自身的水平；3、寻找外界的帮助。4。懂政策技术和指标。

对于这些人群希望通过所谓的技术手段来进行股票的盈利，然而还没有找到比较正确的途径与方法，本系统将服务该部分人群。

### 1.2.4运行环境

(1) 硬件平台

|  |  |  |
| --- | --- | --- |
| 客户端 | | |
| 序号 | 硬件 | 相关组成描述 |
| 1 | PC端客户机 | 能够发送http请求，具备一定的json格式解析能力 |

|  |  |  |
| --- | --- | --- |
| 服务器端 | | |
| 序号 | 硬件 | 相关组成描述 |
| 1 | 路由器 | 可以稳定发送信息和连接网络的设备 |
| 2 | 服务器 | 暂时进行考虑中，使用云平台或直接使用家用电脑作为主要平台，最低配置要求为：英特尔至强系列或i3-8100或以上，内存8GB以上，硬盘空闲1GB以上，带宽1Mb/s以上 |

(2) 软件平台

1、Docker：容器（进阶目标）

2、MongoDB: 数据库

3、JDK1.8或OpenJDK1.8以上

### 1.2.5外部接口需求

(1)第三方股票查询接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口描述 | 所需服务 | 数据和控制信息 | 通信定义 | 需求来源 |
| 网易股票接口 | 查询股票信息 | 所有需要读写的数据 | 第三方接口 | 需要获取股票的历史数据与部分的参数 |

### 1.2.6功能需求

(1) 查询股票历史数据

返回数据必须包含股票的当日基本信息，能够根据日期进行必要的查询，参考sina股票接口数据。

(2) 查询上市公司的基本信息

返回数据必须包含该上市公司的基本信息，其中包含（行业板块信息，公司所在地等与法律相关信息，以及股票分析所需的基本信息，具体请参见雪球公司信息）

(3) 查询其他技术指数信息

返回数据包含股民希望知道的一些指数技术信息以及一部分的网络信息（评论数，关注度等等）

(4) 预测信息

返回数据需要包括提供的预测信息，和建议购买的股票编号与推荐程度，用户可以通过结合自身的经验与本程序提供的各种建议与推荐程度，进行一个有选择性的投资。

并且对于该接口希望能够获取不同算法下的推荐值，或者根据不同的算法进行的结果整合也是需要的。

(5) 业务场景

假设现在有一名无丰富股票基础知识与投资经验的用户A，一位刚刚进入市场，懂得一些股票知识，但是并不熟悉股票的真正运作方式，并且没有过的多的股票投资经验的用户B，以及一位有多年股票投资经验并且对于股票有很深的了解无论从政策方面还是经济方面对于个股有着自身的理解的用户C。以上三位用户都希望通过股票来进行合理的投资是自己的财富或者生活更加美好或有趣。

对于A用户，一位对股市没有过多了解的人来说，希望得到一款系统，能够直接的向这名用户指明那些股票是值得购买的，而这些股票中最推荐购买的又是哪一种，对于这样的完全新手型的股票购买者来说程序需要极其的简单，简单到可以直接不需要通过任何思考就能获利，并提供一个比较便利的途径来达成目的，他只需要一个接口就是最终结果。

对于B用户，一位对于股票有一定的了解，但是无法总是做出正确的决定的投资者而言，他希望调用一个预测接口，并且对于股票添加一些简单的判断，他会调用多种算法实现的不同的股票预测结果，然后再对于每一个股票调用，指数型信息，来实现自己的简单判断，然后在自己的脑海或其他方式中总结出，哪一支股票可以帮助到他，也就是，他需要的简单指数信息和不同算法下的预测结果信息与推荐程度。

对于C用户而言，作为一个资深的股民，他需要一个程序来帮助他缩小范围，能够更加精准快速的找出具有良好潜力的股票，来节约大量的时间，同时他不需要太多的基础技术型指标来帮助他进行判断，他会打开其他比较熟悉的软件进行额外的判断。

(6) 功能需求

优先级：5为最高，1为最低，数值越大将会越优先完成

|  |  |  |
| --- | --- | --- |
| 用户 | | |
| 需求名称 | 优先级 | 备注 |
| 查询股票历史数据 | 3 |  |
| 查询上市公司相关数据 | 3 |  |
| 查询其他技术指标信息 | 2 |  |
| 查询推荐购买的股票 | 5 | 需要使用多种模型进行推理啊 |

|  |  |  |
| --- | --- | --- |
| 管理员 | | |
| 需求名称 | 优先级 | 备注 |
| 爬去信息接口 | 5 | 需要爬去各种信息，并且完成信息的存储 |

### 1.2.7非功能性需求

A 性能需求

1、数据精准度

时间精准到天

由于部分股票价格剥夺比较薄弱，使用股票价格精准至分

2、并发特性

在网络稳定并且服务器达到标准时，可以承受大于1000的并发量并且不产生丢包或数据丢失等情况

3、故障处理特性

保证在每十五秒为周期检查接口活跃度，如果接口为非活跃状态或假死状态则停止服务并自动重新启动

B 安全措施需求

1、程序将会拥有一套完整的日志体系，可以打印出所有的错误发生和时间点，并且通过接口活跃度检查，将会进行邮件通知的报警行为。

2、 应用日志必须按日保存，每个文件保存应用一天的完整日志；

3、 历史应用日志必须压缩后按日保存，以减小历史日志文件对硬盘空间的占用；

4、 历史应用日志必须保留一个月（30天），以保证问题出现后，日志可以得到较好的保存。

C安全需求

系统要有足够的防御能力，防御普通黑客的攻击，尽量减少各种可能出现的漏洞，防止用户信息被不良分子窃取。

保证用户权限的正确设置，防止出现各种越权行为，保证系统的正常良好运行。用户的信息保存采用国际流行的MD5方式加密，虽然现在有了许多MD5密码库，但是这还是不失为保证安全的一种好方法。

管理员接口将进行简单的鉴权（输入一串加密字符串，并进行解析鉴权）

D 操作需求

系统操作者多为非专业人士，所以必须提供直观的图形界面，使能够快速上手使用。

E 质量需求

易用性优于易学性，或者可移植性优于有效性。

### 1.2.8软件需求规格表

|  |  |  |  |
| --- | --- | --- | --- |
| 软件需求表 | | | |
| 子系统 | 模块内容 | 对应接口 | 功能说明 |
| 日志与测试系统 | 日志分析  自动化测试 | 日志分析接口  自动化测试检测接口 | 获取与分析日志内容，以最快的速度对日志进行分析，并且对错误日志进行汇总并且提交邮件，同时进行接口活跃度测试，实时监控接口活跃度达到良好的用户体验 |
| 安全系统 | 基本鉴权 | 包装型的鉴权接口 | 对于管理员的请求进行轻量级的鉴权服务，以此来抵挡误输入与部分的攻击行为，并提供给数据以基本的安全体系 |
| 核心算法预测系统 | 股票推荐与分数排名 | 各算法对应的推荐接口 | 本程序的核心算法与核心需求，对于本部分的接口我们使用不同的算法进行分析，以此来保证准确性。 |
| 爬虫系统 | 将存在于网络的上的数据接入本地或远程数据库 | 网易爬虫接口  百度爬虫接口  雪球爬虫接口  东方财富爬虫接口  新浪爬虫接口  官方爬虫接口 | 对于不同的数据我们使用不同的数据源来进行良好的爬去，对于一些特定的数据源我们也将使用特定的爬虫以此获取，这个系统也同样是本程序的核心算法之一，因为所有的数据分析都是建立在已有数据的基础上的 |
| 数据展现系统 | 将数据库包含的数据展示给用户 | 历史记录接口  技术指标接口  公司信息接口 | 主要提供给第三方或股民用户其他的评判条件 |

# 2 概要设计说明书

## 2.1引言

本部分内容将会简介的告诉开发者与读者，我们的程序将如何运作，如何设计，层次以及包含的大致内容。在阅读完本章节的内容后，我们希望读者能够自行想象出一套大致的框架，在之后的详细设计中我们才会详细的告诉你这一套框架中的具体内容与实现方法，为了能够使读者更加清晰的了解到我们表达的概念，我们会使用图形，表格等方式进行说明。

在阅读本篇章之前，希望读者能够快速的或详细的阅读上文的需求分析，在阅读完本篇章之后，我们会对总体结构有一定的解释与说明，我们试着回答一下这些问题：我们将会以哪种形式完成这份程序；我们将会以哪种方式完成这些函数；哪些东西是必要的；程序的总体结构是什么以及哪些数据是必要的，并了解这些数据的含义。

当然还有很多东西是读者无法从本篇章了解的，例如具体实现逻辑，参与运算的具体参数，等等，这些东西将会在详细设计中说明，而本篇章将不会赘述，也不会纠缠。

同时我们将在详细设计的对本文档不断地进行细节更改，所以在阅读之前请确定，本篇章为最终版本。

## 2.2 总体设计

### 2.2.1需求

使用户获得股票信息与预测信息

### 2.2.2目标

通过程序预测使用户能够快速有效的盈利

### 2.2.3运行环境

OpenJDK1.8及以上，MySQL8.0.3及以上，Mongo3.0及以上

### 2.2.4局限

本程序只提供概率性与大致的预测，无法提供极为准确的预测，并且由于部分原因，无法提供详实的数据，所以无法满足一些特殊需求，也会发对特定条件进行针对性分析。

### 2.2.5结构

本程序因为功能简单，并没有使用时下最为流行的微服务架构，如果改为微服务架构，会降低耦合，但是会增大代码的冗余，并且增加运行成本，本程序暂时不考虑使用。

本程序使用经典的三层架构：dao层，service层，controller层，十分中庸的解决了现在问题，并且作为最经常被使用的架构，它有着与众不同的简单和稳定。

## 2.3.接口设计

### 2.3.1对外接口

外部接口均以utf-8作为标准编码，并且对于各个接口的返回数据均以JSON格式作为标准返回格式，其余规定将在一下表格中进行备注说明。同时也因为本项目暂时没有考虑安全性问题，所以并没有类似于authorization的鉴权手段。

数据展示模块

一、K线图接口

a. 主要实现功能

本接口将返回某股票自1996年至今的所有走势，同时在调用者调用正确的日期时，也需要提供某一时间段内的价格走势图

b. 主要的传入参数

主要传入参数为开始日期，截止日期，以及正确的股票编号。

c. 主要的返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数含义 | 类型 | 示例数据 | 备注 |
| img | 图片 | base64 | - | - |

d. 额外的说明与备注

本接口为get方法传参，并且当用户出入非开盘日期或传入的该股票处于停牌阶段的我们将向前选择就近的日期并计算出相应的值，单用户只需要简单的json数据时，本接口也可以做一定的拓展。

二、上市公司信息接口

a. 主要实现功能

本接口主要返回用户想要知道的一些关于公司的基本信息，如所在地，注册资本等等，我们希望单用户使用本接口并获得数据后能够对该上市公司有一个比较大致的了解。

b. 主要的传入参数

传入参数主要为上市公司的股票代码。

c. 主要的返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数含义 | 类型 | 示例数据 | 备注 |
| 公司名称 | 公司的实际名称 | String | 浙江广厦股份有限公司 | - |
| 公司名称2 | 公司名称简称 | String | 浙江广厦 | - |
| 成立日期 | 公司的成立日期 | String | 19930713 | - |
| 注册资本 | 注册资本 | String | 87178.91 | - |
| 地区名称 | 公司所在地区的名称 | String | 浙江省 | - |
| 地区编号 | 公司所在地区的编号 | String | CN330000 | - |
| 一级板块名称 | 名称 | String | CN330000 | - |
| 一级板块编号 | 编号 | String | 1102 | - |
| 所属行业名称 | 二级行业所属的名称 | String | 房地产开发 | - |
| 所属行业编号 | 所属的行业编号 | String | 430100 | - |

d. 额外的说明与备注

本接口同样使用的是get方法进行参数的传递。

三、股票单日价格查询接口

a. 主要实现功能

此接口为股票的单日价格接口，输入日期和股票代码后，我们将会从数据库中查找相对应的股票信息，同时提供多种数据的展示。

b. 主要的传入参数

比较重要的数据为正确的日期，与正确的股票代码，当为输入正确的参数时，我们将返回空值，或错误信息。

c. 主要的返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数含义 | 类型 | 示例数据 | 备注 |
| 日期 | 交易日当天的日期 | String | 20090313 | - |
| 开盘价 | 交易日当天的开盘价格 | Double | 3.95 | - |
| 收盘价 | 交易日当天的收盘价格 | Double | 3.87 | - |
| 浮动金额 | 单日最高价与最低价的差值 | Double | -0.08 | - |
| 浮动比例 | 浮动金额的比例 | Double | -2.03 | - |
| 单日最低价 | 交易日当天的最低价格 | Double | 4.03 | - |
| 单日最高价 | 交易日当天的最高价格 | Double | 3.86 | - |
| 成交量 | 交易日当天该股票的成交总量 | Double | 629.36 | - |
| 成交额 | 当天的成交总额 | Double | 2472.21 | - |
| 换手率 | 股票交换的比例 | Double | 0.82 | - |

d. 额外的说明与备注

本方法为get请求，通过url传值。

四、上市公司经济参数接口

a. 主要实现功能

返回和提供上市公司的主要经济参数，希望用户通过调用本接口，能够对于希望了解的上市公司的经济情况有一个大致的了解与研究，但是本程序只提供最为基础的数据信息，并不包含更加深入的类似于证券交易所的收费查询信息，希望注意。

b. 主要的传入参数

如上一接口，本接口同样需要上市公司的股票代码来进行查询，但是不需要时间。

c. 主要的返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数含义 | 类型 | 示例数据 | 备注 |
| 日期 | 报表日期 | String | 20171231 | - |
| 每股收益 | 每股的净收益 | Double | 0.860 | - |
| 每股净资产 | 每股所有拥有的净资产 | Double | 7.381 | - |
| 每股现金流 | 请参考数据词典 | Double | 0.309 | - |
| 净资产增长率 | 请参考数据词典 | Double | 9.732 | - |
| 净资产收益率（加权） | 请参考数据词典 | Double | 12.240 | - |
| 主营业务收入增长率 | 请参考数据词典 | Double | 17.481 | - |
| 净利润增长率 | 请参考数据词典 | Double | 111.222 | - |
| 总资产增长率 | 请参考数据词典 | Double | -2.46 | - |
| 销售毛利率 | 请参考数据词典 | Double | 14.067 | - |
| 主营业务收入 | 请参考数据词典 | Double | 289092900259.280 | - |
| 主营业务利润 | 请参考数据词典 | Double | 24924170426.650 | - |
| 利润总额 | 请参考数据词典 | Double | 24035130135.610 | - |
| 净利润 | 请参考数据词典 | Double | 19170337569.640 | - |
| 资产总额 | 请参考数据词典 | Double | 350234632615.17 | - |
| 负债总额 | 请参考数据词典 | Double | 175762228075.96 | - |

d. 额外的说明与备注

GET方法作为主要的方法，该接口只会返回最新的财务报表，其中财务报表中也会有一定的缺省值，有可能来自于上市公司的统计不足，并不是接口问题。

(2) 数据预测模块

一、股票chart计算接口

a. 主要实现功能

作为本项目核心接口，该接口主要为用户表现了，但股票处于上涨与下跌是，经过一定的计算，他们对于某一相同的随机概率算法，但是展示出了不同的统计学表现与不同的概率分布，那么就能够说明这个曲线对于股票的上涨与下跌有一个指示性的作用。同时也就能够证明了理论的成立。

b. 主要的传入参数

要得出入参数只有一个运算量，也就是一个数字。

c. 主要的返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数含义 | 类型 | 示例数据 | 备注 |
| img | 以url或Base64作为资源的图片资源 | String | locahhost:9902/static/chartImg/… | 涨跌两条曲线分布 |

d. 额外的说明与备注

因为受限于计算机的性能，当作该接口设计时我们应该考虑到该接口的极限运算能力，与计算时间，所以显而易见的是该接口的运算量会有一个最大值，或者说有一个极限值，同时这个接口但运算次数过少的时候无法真实的体现出股票涨跌的规律，使用理论上应该也有一个最小值，使用希望在之后的实际代码编写过程中能够通过经验和实际数据，能够获取这个接口的最大值或最小值。故在此不做过多的说明与解释。

二、股票chart预测接口

a. 主要实现功能

作为本项目的核心接口之一，该接口主要的目的是上一接口运算所得的结果进行一个合成，也就是说本接口才是真正的预测接口。本接口我们将提供出一张拥有三条曲线的图片，分别为上涨曲线，下跌曲线，本股票的曲线，用户可以根据我们提供出的图标数据自己进行判断。

b. 主要的传入参数

与上一接口相同的运算量，以及他希望预测的股票编号。

c. 主要的返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 股票chart预测接口 | | | | |
| 参数名称 | 参数含义 | 类型 | 示例数据 | 备注 |
| img | 以url或Base64作为资源的图片资源 | String | locahhost:9902/static/chartImg/… | 希望三条曲线能够合成一张图 |
| 涨跌幅度 | 该股票的涨跌幅度 | Double | 17.334 | - |

d. 额外的说明与备注

本接口与上一接口拥有相同的特点，也就是拥有一个极限运算量，与最小运算量，同时在调用本接口之前我们建议用户先调用上一接口，因为本接口将调用上一接口的运算数据，如果没有上一接口的运算数据，我们将返回错误给用户，使用请注意。

三、数学公式预测接口

a. 主要实现功能

主要实现功能为给出一个股票未来上升或下降的数值。

b. 主要的传入参数

股票代码。

c. 主要的返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数含义 | 类型 | 示例数据 | 备注 |
| 股票上升幅度 | 股票在一定时间内的上升幅度 | Double | 1.924 | - |

d. 额外的说明与备注

与之前的数学公式图标预测相同，都是通过大量的测试数据进行推导得出的，使用单计算量越大越能符合真实情况。

### 2.3.2内部接口

本内容与上文部分内容重合，特别是体现在返回参数上，使用并且该部分内容需要开发者自行考虑，此说明文档主要起到一个提示作用，并不会具体详细的解释各个参数，而是将接口内容大致说明，具体参数请由开发者自身决定。

## 2.4 功能模块设计

### 2.4.1爬虫模块

本模块均为入库操作，只存在success或fail两种返回值。在操作本模块的时候请一定注意，部分接口有可能存在访问限制，所以请谨慎使用，并且考虑到合理的共存，请不要为了追求性能使用过多的线程，以此给他人服务器参数大于预测的压力。

一、东方财富网爬虫接口

抓取东方财富网的股票列表信息，并且进行解析，并且将结果入库，在此我们希望获得两个数据，即股票代码，以及股票所属交易所，并且只抓取上证与深证的主要股票，不包含其他内地小众股票。

二、雪球爬虫接口

抓取雪球网的数据，以及雪球网上所提供的公司数据与企业报表数据。该接口经过预研发现可以使用。

三、搜狐爬虫接口

历史股价信息。

四、百度手机爬虫接口

股票基础数据。

### 2.4.2基础数据服务模块

一、原始数据转换一层数据接口

将上文中所提及的抓取获得的原始数据直接转化为一级数据格式，也就是进行一些处理，我们需要的是提炼出比较关键的参数，并且对这些参数进行格式化，具体的参数计算以及参数的选取我们会在详细设计中严格的说明，但是在这里我们这是提及我们需要这些东西，来为下一步的二级数据做一个准备。

二、一层数据转二层数据接口

当我们拥有一个一级数据时，我们会发现一级数据已经拥有了一个天然的权重，并且这样的现象是十分严重的，我们无法保证在进行股票比较时，这些权重是否会对最终的结果造成不必要的影响，使用二级数据我们需要将这个天然的权重淡化，甚至于抹平这个权重带来的不必要的影响，使用我们需要对以及数据进行所谓的平等化、去权重化的操作，也就是生成我们的二级数据，如上一接口一样，如何才能去除权重，我们将会在详细设计中较好的诠释出来，使用请阅读下面的篇章。

三、随机数生成接口

随机数生成模块是预测的重要接口，我们需要大量的假想数据，同时我们也需要大量的为结果提供概率分布的数据，而随机数生成接口必须完成这样的任务，也就是使预测结果可统计，并且使其符合统计学规律，所有的计算类的包含具体公式与算法的接口，我们都会在详细设计中给予说明。

### 2.4.3数据预测计算模块

一、公式预测接口

公式预测接口，这不是一个对外开放的接口，这个接口主要通过大量的预测性的运算，并通过一种比较简单的自我学习，自我调整的方式，来进行不断的比较，拟合，同时删除无法进行回归的数据，我们也不在这里讨论十分详细的运算法则，与最终的公式实现。

## 2.5 数据库设计

数据库这里使用的是MySQL与Mongo，日后还会考虑使用PostgreSQL，以及部分的newSQL，对于现有逻辑，MySQL主要存放大批量的结构化数据，或者说可转化为结构化的数据，这些数据需要包含以下特点，需要大量的读取写入，将会存放大量的数据（超过2GB，当mongo类似的内存型数据库超过一定量时，读写效率会大幅度下降），对于本系统，Mongo数据库将存放非结构化数据。

同时因为本项目不包含过多的数据库关系逻辑概念，所以我们将不会对于数据库概念设计E-R图有描述，仅仅只会告诉开发者哪些表是被需要的，以及这些表包含的字段所代表的意义（也就是我们所说的数据字典），同时我们也将对数据库的物理结构进行说明，但是因为是小型数据库，我们暂时不会采用主从数据库类型的分布式结构。

### 2.5.1数据库结构设计

(1) MySQL部分

该部分主要存储结构化数据，即硬数据，不会轻易改变，适合大量读写。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | | | | |
| 股票代码与所属证券交易所 | | | | |
| 字段 | 描述 | 数据类型 | 约束 | 主键 |
| stock\_code | 股票代码 | varchar(30) | 不可为空 | 1 |
| exchange | 股票所属证券交易所 | varchar(30) | - | 0 |
|  |  |  |  |  |
| basic\_info | | | | |
| 上市公司基础信息 | | | | |
| 字段 | 描述 | 数据类型 | 约束 | 主键 |
| code | 股票编号 | varchar(30) | 不可为空 | 1 |
| boardcode | 板块编号 | varchar(100) | - | 0 |
| boardname | 地区代码(省级) | varchar(100) | - | 0 |
| comp\_name | 公司名称 | varchar(100) | - | 0 |
| compsname | 公司股票简称 | varchar(100) | - | 0 |
| found\_date | 成立日期 | varchar(100) | - | 0 |
| keycode | 地区编号 | varchar(100) | - | 0 |
| keyname | 地区名称 | varchar(100) | - | 0 |
| level2code | 二级行业编号 | varchar(100) | - | 0 |
| level2name | 二级行业名称 | varchar(100) | - | 0 |
| regcapital | 注册资本 | varchar(100) | - | 0 |
|  |  |  |  |  |
| fin\_info | | | | |
| 上市公司季度财务报表 | | | | |
| 字段 | 描述 | 数据类型 | 约束 | 主键 |
| id | id号 | varchar(100) | 不可为空 | 1 |
| basiceps | 基本每股收益 | double | 不可为空 | 0 |
| code | 股票编号 | varchar(100) | - | 0 |
| main\_businc\_grow\_rate | 主营业务收入增长率(%) | double | 不可为空 | 0 |
| mainbusiincome | 主营业务收入 | double | 不可为空 | 0 |
| mainbusiprofit | 主营业务利润 | double | 不可为空 | 0 |
| naps | 每股净资产 | double | 不可为空 | 0 |
| netassgrowrate | 净资产增长率(%) | double | 不可为空 | 0 |
| netinc\_grow\_rate | 净利润增长率(%) | double | 不可为空 | 0 |
| netprofit | 净利润 | double | 不可为空 | 0 |
| o\_per\_cash\_per\_share | 每股现金流 | double | 不可为空 | 0 |
| report\_date | 报表日期 | varchar(100) | - | 0 |
| salegrossprofitrto | 销售毛利率(%) | double | 不可为空 | 0 |
| totalassets | 资产总额 | double | 不可为空 | 0 |
| totalliab | 负债总额 | double | 不可为空 | 0 |
| totassgrowrate | 总资产增长率(%) | double | 不可为空 | 0 |
| totprofit | 利润总额 | double | 不可为空 | 0 |
| weightedroe | 净资产收益率(加权)(%) | double | 不可为空 | 0 |
|  |  |  |  |  |
| hq\_info | | | | |
| 股票历史价格 | | | | |
| 字段 | 描述 | 数据类型 | 约束 | 主键 |
| id | id号 | varchar(100) | 不可为空 | 1 |
| close | 收盘价 | double | 不可为空 | 0 |
| code | 股票编号 | varchar(100) | 索引 | 0 |
| data | 报表日期 | varchar(100) | - | 0 |
| exchange\_rate | 换手率（%） | double | 不可为空 | 0 |
| float\_pre | 浮动比例（百分比） | double | 不可为空 | 0 |
| floatw | 浮动金额 | double | 不可为空 | 0 |
| high\_price | 最高 | double | 不可为空 | 0 |
| low\_price | 最低 | double | 不可为空 | 0 |
| open | 开盘价 | double | 不可为空 | 0 |
| vol | 成交量（万） | double | 不可为空 | 0 |
| vol\_price | 成交额（万） | double | 不可为空 | 0 |
|  |  |  |  |  |
| snap\_shot | | | | |
| 股票实时简单经济指数 | | | | |
| 字段 | 描述 | 数据类型 | 约束 | 主键 |
| code | 股票编号 | varchar(100) | 不可为空 | 1 |
| bv\_ratio | 市净率 | double | 不可为空 | 0 |
| circulating\_capital | 流通股 | int | - | 0 |
| net\_assets\_per\_share | 每股净资产 | double | 不可为空 | 0 |
| per\_share\_earn | 每股收益 | double | 不可为空 | 0 |
| peratio | 市盈率 | double | 不可为空 | 0 |
| total\_share\_capital | 总股本 | int | - | 0 |
|  |  |  |  |  |
| first\_calculation\_unit | | | | |
| 一层计算单元 | | | | |
| 字段 | 描述 | 数据类型 | 约束 | 主键 |
| code | 股票编号 | varchar(100) | 不可为空 | 1 |
| bv\_ratio | 市净率 | double | 不可为空 | 0 |
| fifteen\_price | 15日均价 | double | 不可为空 | 0 |
| four\_hundred\_price | 400天均价 | double | 不可为空 | 0 |
| history\_price | 历史均价 | double | 不可为空 | 0 |
| low\_fifteen\_price | 15日最低价 | double | 不可为空 | 0 |
| low\_four\_hundred\_price | 400天最低价 | double | 不可为空 | 0 |
| low\_history\_price | 历史最低价 | double | 不可为空 | 0 |
| low\_one\_eighty\_price | 180天最低价 | double | 不可为空 | 0 |
| low\_sixty\_price | 60日最低价 | double | 不可为空 | 0 |
| low\_thirty\_price | 30日最低价 | double | 不可为空 | 0 |
| mainbusiincome | 主营业务收入 | double | 不可为空 | 0 |
| mainbusiincome\_avg | 历史主营业务收入平均值 | double | 不可为空 | 0 |
| netinc\_grow\_rate | 净利润增长率(%) | double | 不可为空 | 0 |
| netprofit | 净利润 | double | 不可为空 | 0 |
| netprofit\_avg | 历史净利润平均值 | double | 不可为空 | 0 |
| now\_price | 时价 | double | 不可为空 | 0 |
| one\_eighty\_price | 180天均价 | double | 不可为空 | 0 |
| peratio | 市盈率 | double | 不可为空 | 0 |
| sixty\_price | 60日均价 | double | 不可为空 | 0 |
| thirty\_price | 30日均价 | double | 不可为空 | 0 |
| totalassets | 资产总额 | double | 不可为空 | 0 |
| totalassets\_avg | 历史资产总额平均值 | double | 不可为空 | 0 |
| up\_rate | 60日涨幅(%)(1/4 \* 前30日 + 3/4 \* 后半段30日均价) | double | 不可为空 | 0 |
|  |  |  |  |  |
| second\_calculation\_unit | | | | |
| 二层计算单元 | | | | |
| 字段 | 描述 | 数据类型 | 约束 | 主键 |
| code | 股票编号 | varchar(100) | 不可为空 | 1 |
| bv\_ratio | 1 / ( 1 + e^( -市净率 / 市净率平均值)) | double | 不可为空 | 0 |
| fifteen\_price | 15日均价 / 时价 | double | 不可为空 | 0 |
| four\_hundred\_price | 400天均价 / 时价 | double | 不可为空 | 0 |
| history\_price | 历史均价 / 时价 | double | 不可为空 | 0 |
| low\_fifteen\_price | 15日最低价 / 时价 | double | 不可为空 | 0 |
| low\_four\_hundred\_price | 400天最低价 / 时价 | double | 不可为空 | 0 |
| low\_history\_price | 历史最低价 / 时价 | double | 不可为空 | 0 |
| low\_one\_eighty\_price | 180天最低价 / 时价 | double | 不可为空 | 0 |
| low\_sixty\_price | 60日最低价 / 时价 | double | 不可为空 | 0 |
| low\_thirty\_price | 30日最低价 / 时价 | double | 不可为空 | 0 |
| mainbusiincome | 1 / ( 1 + e^( -主营业务收入 / 历史主营业务收入平均值)) | double | 不可为空 | 0 |
| mainbusiincome\_avg | 历史主营业务收入平均值 | double | 不可为空 | 0 |
| netinc\_grow\_rate | 1 / ( 1 + e^( -净利润增长率 / 净利润增长率平均值)) | double | 不可为空 | 0 |
| netprofit | 1 / ( 1 + e^( -净利润 / 历史净利润平均值)) | double | 不可为空 | 0 |
| netprofit\_avg | 历史净利润平均值 | double | 不可为空 | 0 |
| now\_price | 时价 | double | 不可为空 | 0 |
| one\_eighty\_price | 180天均价 / 时价 | double | 不可为空 | 0 |
| peratio | 1 / ( 1 + e^( -市盈率 / 市盈率平均值)) | double | 不可为空 | 0 |
| sixty\_price | 60日均价 / 时价 | double | 不可为空 | 0 |
| thirty\_price | 30日均价 / 时价 | double | 不可为空 | 0 |
| totalassets | 1 / ( 1 + e^( -资产总额 / 历史资产总额平均值)) | double | 不可为空 | 0 |
| totalassets\_avg | 历史资产总额平均值 | double | 不可为空 | 0 |
| up\_rate | 60日涨幅(%)(1/4 \* 前30日 + 3/4 \* 后半段30日均价) | double | 不可为空 | 0 |

(2) Mongo部分

此部分为非结构化数据，并且只会存放临时数据，在这里只会表达出关键词语的含义，并不会做过多的解释。

### 2.5.2数据库环境

Windows10操作系统，MySQl5.6或以上，Mongo3.3.x或以上

## 2.6 程序模型设计

### 2.6.1程序主流程

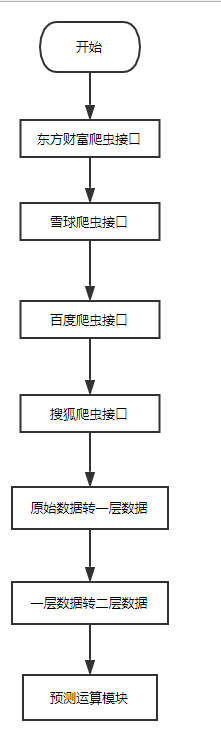


图5-1-1

### 2.6.2 运行预测与安全

(1) 运行时间与所占用资源

以下标准由具体环境变动而变动，此处估计时间为中上级家用台式机运行速度。

爬虫模块：总占用资源 磁盘 4GB

东方财富网爬虫：30秒以内

雪球爬虫：30分钟以内

百度爬虫：10分钟以内

搜狐爬虫：60分钟以内

数据处理模块：占用资源 磁盘50MB

原始数据转一级数据：10分钟以内

一级数据转二级数据：20分钟以内

数据计算模块：占用资源 磁盘20GB以下

图表计算：按计算量而定，一次计算应控制在8个小时以内

公式计算：同上

预测模块：无占用

图片预测：同上

公式预测：同上

总运行时间与占用资源估计

硬盘资源：25GB以下

主流程运行时间不超过24个小时

### 2.6.3安全与备灾

(1) 系统安全

因为不存在用户鉴定，使用不会设置必要的用户鉴权，但是因为需要防止爬虫的侵害，我们希望开发者设置一个简单的IP过滤器，做一个简单的过滤，在详细设计中我们会具体介绍这个IP过滤器的具体实现方法。

(2) 系统备灾

无

# 3系统详细设计

## 3.1 引言

本部分内容，我们将详细的说明之前在以上概要设计模块中提及的接口实现方法，以及具体的实现方式，我们有可能通过各种方法来进行说明，如标准流程图，伪代码等等方式，我们旨在更加明确地告诉程序编写人员，如何才是我们真正想要的逻辑，以及实现我们上述所讲的这些接口的方法与中间过程。

## 3.2 定义

我们将描述以下具体说明中有可能具体会被使用的技术或者难以理解的词汇，如果编程人员对一下词汇已经十分熟悉，可以跳过本部分的阅览。

1. Spring Data JPA

一种基于表结构的数据访问组件。

二、RestTemplate

**客户端HTTP访问的中心类。**它简化了与HTTP服务器的通信，并实施了RESTful原则。它处理HTTP连接，让应用程序代码提供URL（带有可能的模板变量）并提取结果。

三、谷歌Java风格指南

由谷歌公司发布的一种Java编写规范，其中规定了许多必要的规则，如命名等等，同时也是本程序主要参考的规则之一，<https://google.github.io/styleguide/javaguide.html>

四、logistic回归

logistic模型（或Logit模型）是一个统计模型，通常被应用于二元因变量。在回归分析，逻辑回归或分对数回归被估计逻辑模型的参数。更正式地说，一个逻辑模型就是一个事件的概率的对数几率是线性组合的独立或预测变量。两个可能的因变量值通常被标记为“0”和“1”，它们表示诸如合格/不合格，胜负/失败，活着/死亡或健康/生病等结果。二元逻辑回归模型可以推广到两个以上的因变量水平：具有多于两个值的分类输出通过多项逻辑回归建模，并且如果多个类别是有序的，则通过有序逻辑回归，例如比例几率序数逻辑模型。

六、股票交易所

SH代表上海证券交易所，SZ代表深圳证券交易所。

## 3.3整体框架与规范

### 3.3.1整体框架

本程序使用SpringBoot作为快速开发的框架，在数据库读写方面会使用SpringDataJPA对MySQL进行访问，而对于Mongo的访问与读取请使用最为普通的连接数据库进行读写的方式，RESTful网络访问使用restTemplate。

### 3.3.2规范

全部编写工作请遵行Google java编写指南。

## 3.4 程序描述

### 3.4.1总体包图

包类介绍

Crawler：爬虫模块

DBRepository：JPA数据存储模块

Forecast：数据预测模块

MongoDB：mongo数据库操作模块

CalculationUnit：数据计算模块

DataShow：数据展示模块

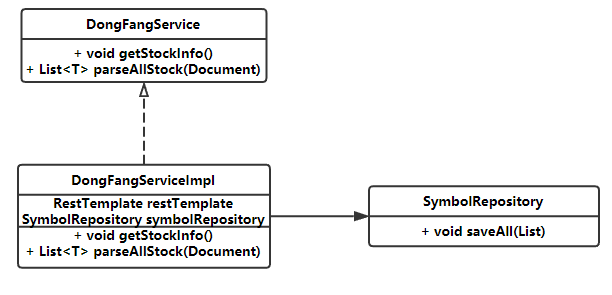
Controller：对外接口控制层

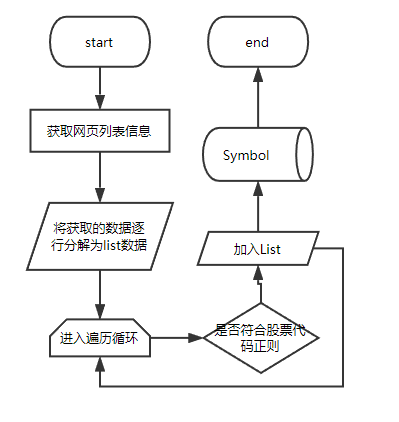
一下将会对各个类包中的具体实现方式进行说明。

### 3.4.2爬虫模块

(1) 东方财富网爬虫

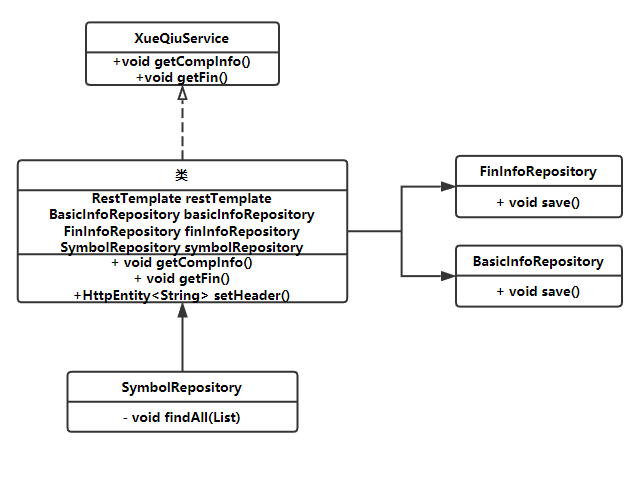
抓取东方财富网中的股票代码数据。在入库的数据中请检查他是上证所的或是深圳所，并加入对应的标识。

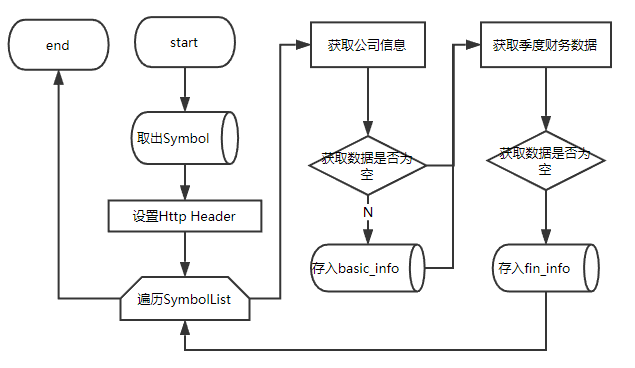




(2) 雪球爬虫模块

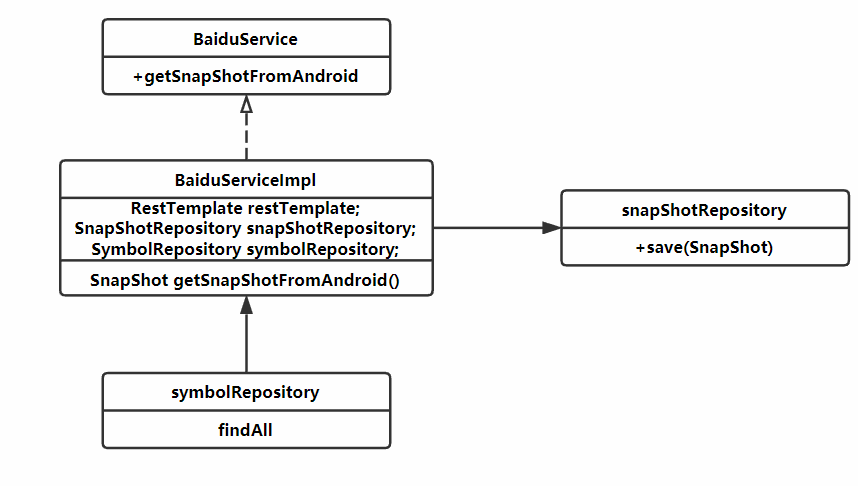
对于雪球的爬虫，我们需要访问雪球的数据接口并进行爬取，在爬取的过程应该会用到Cookie，使用请注意，这里的setHeader()方法就是将Cookie插入Https访问中，请注意。

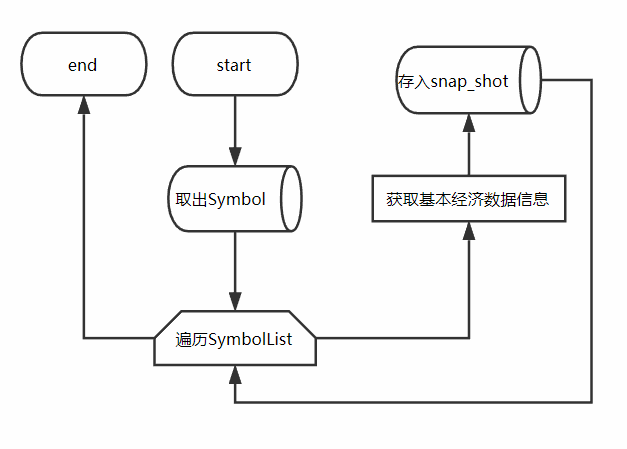




(3) 百度爬虫接口

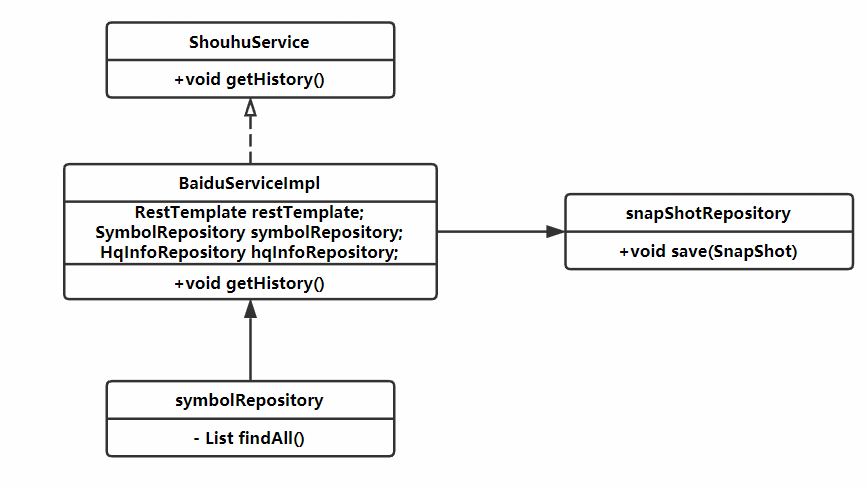
百度爬虫接口主要爬取一些实时的经济数据。

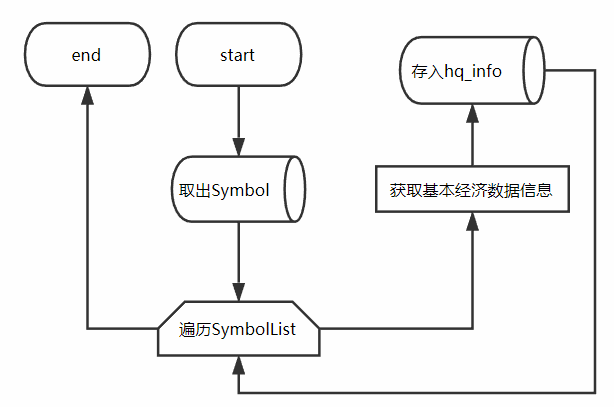




(4) 搜狐爬虫接口

抓取K线图等重要数据。



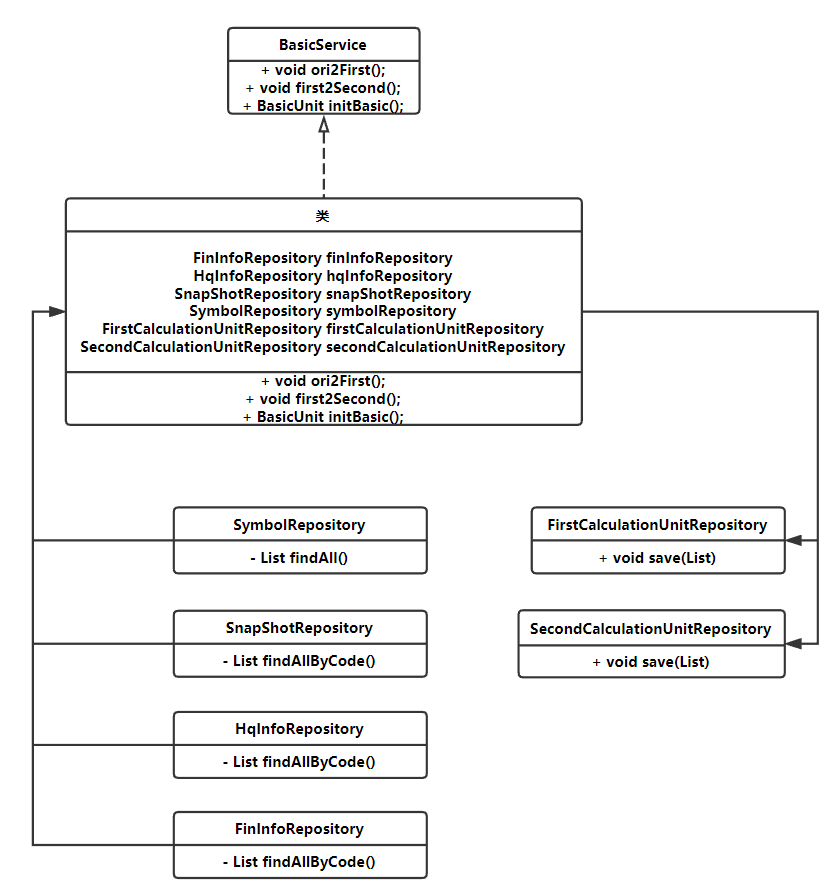


### 3.4.3计算模块

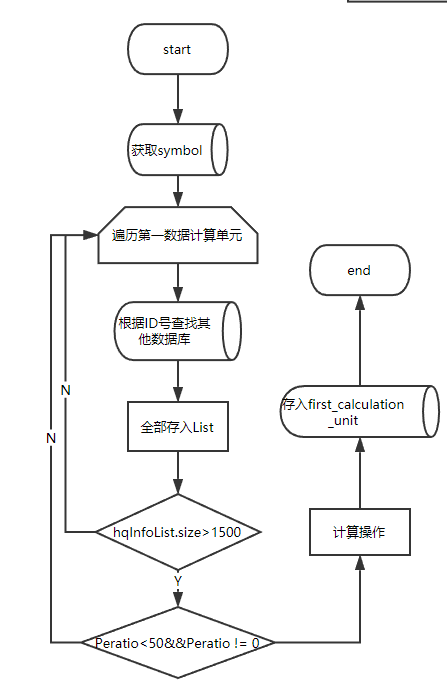
计算模块中会将所有的原始信息进行判断以及检验，首先判断是是否适合进行数据分析，其次我们会在二层数据中将所有数据自带的权值统一，也就是降低原来的值对于最后结果的影响。

在计算过程中我们使用了logistic回归这一概念，通过这个值在一定范围之内的函数，我们可以将所有的数据尽可能的保持在同一高度，也就是将权重降为基本相同。

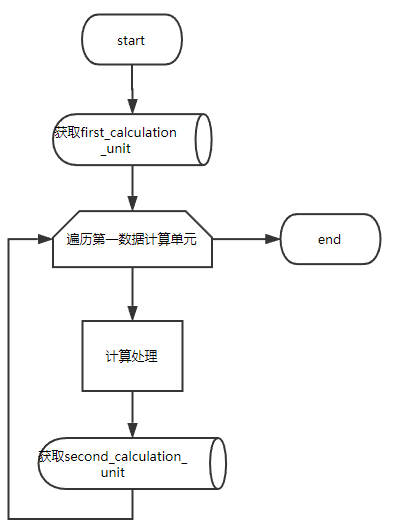
所有公式内容请参考概要设计中数据库设计，此处不做过多陈述。



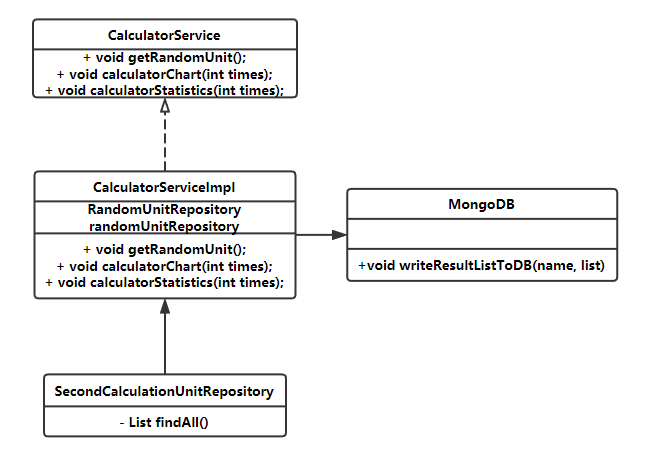
原始数据转一层数据

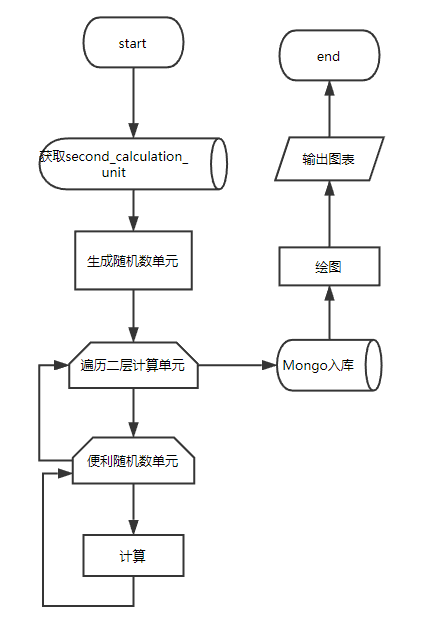


一层数据转二层数据



### 3.4.4计算预测模块





## 3.5限制与部署

以上只列出了主要的核心算法模块，并没有提及数据展示模块，但是这里将会给出限制。

外部访问限制只允许GET与POST方法，所有的DEL方法不能被使用。

# 4 主要技术与关键模块实现

我们将会从主要技术，核心代码两个角度来解析，一整个项目的，并且展示出我们已经实现的效果，对于同时也会展示我们的一些失败的尝试，并且告诉大家为什么我们认为他是失败的。首先引入我们的主要技术。

## 4.1主要技术

(1) SpringBoot与Maven

我相信这个无需多做赘述，这是本程序的一大特色，脱离了大量的配置阶段，上手即可使用，同时也是当下最为流行的JavaEE框架。

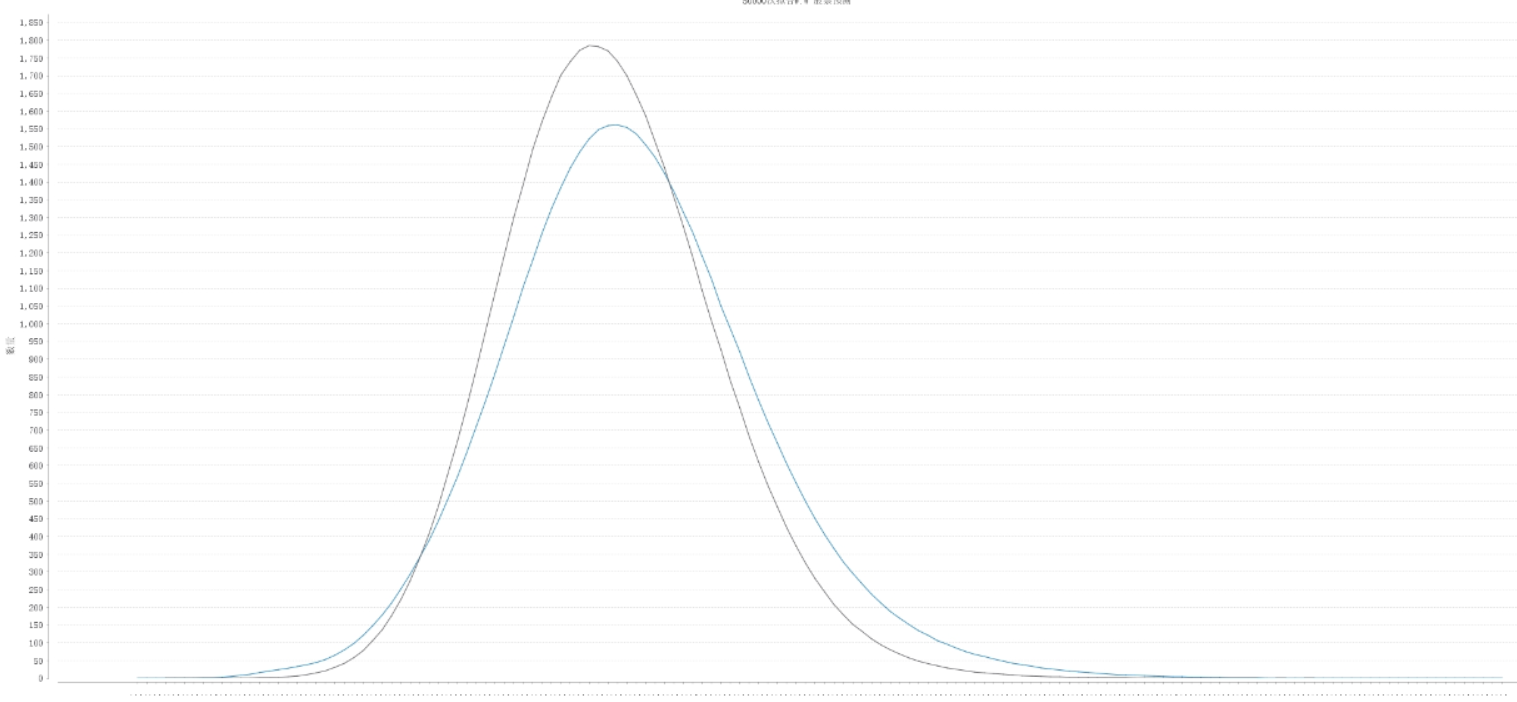
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web-services</artifactId>  
</dependency>

SpringBoot被证明能够适应当下最为流行的快速开发与微服务架构，大量的方法封装将Java反射机制展现的淋漓尽致，同时由于大量的封装也使开发变得更加简单快捷，对于约定大于配置这样的灵活机制，是整个程序又处于高度可定制化，使用了简单的配置文件，虽然现在大量的YML文件被使用，但是本程序还是使用了最为普通的application.properties的多环境配置。YML虽然十分简洁，但是由于一些原因，他并不十分容易被编辑，并且用过空格来进行格式规定，这也使一个小型项目相对于较为原始的方法。

spring.jpa.show-sql = **true**logging.level.org.springframework.data=DEBUG  
spring.jpa.hibernate.ddl-auto=update  
  
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/stock?useSSL=true  
spring.datasource.username=root  
spring.datasource.password=123456  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.data.cassandra.ssl=**true**server.port=9902

(2)Jfreechart工具类

这个工具本来是用于Swing编程的，但是现在被我使用来进行图标的绘制，同时因为我们有一个明确的前端，我也无法使用前端的插件来进行绘制，这个组件拥有大量的可定制化元素，可以绘制，饼图，折线图，柱状图等等大量的图片，对于以上这些东西都有很好的支持，本程序中只使用到了极为细小的一部分内容，折线图，通过这个工具我们得到了我们想要的效果，类似于下图。



这张图并没有展示的十分清晰，我们会在下面的核心代码介绍中详细的介绍一下这张图，但是现在读者只需要知道，这是一张有Jchart所绘画出来的效果，并且这只是一个没有经过美化包装的效果图。

绘画这样的图大致流程就是“输入数据”，“指定样式”，“输出到文件中”

JFreeChart chart = ChartFactory.*createLineChart*(  
 title, xTitle, yTitle, ChartUtils.*categoryDataset*(map));  
ChartUtilities.*saveChartAsJPEG*(  
 new File("img/"+title + ".jpeg"), //文件保存物理路径包括路径和文件名  
 1.0f, //图片质量 ，0.0f~1.0f  
 chart, //图表对象  
 width, //图像宽度 ，这个将决定图表的横坐标值是否能完全显示还是显示省略号  
 height);

(3) JPA

Spring.data一大功能，很好的整合了各个数据库，将所有的数据库真正的实现了统一配置，本程序中使用了MySQL作为主要的存储数据库，在公司的电脑中使用了年度数据库PostgreSQL作为存储数据库，在这个改变的过程中唯一的区别就是将数据库配置进行改变而核心代码几乎没有任何的改变就能够直接被使用，这就是统一配置的最为方便的地方，以下我们会暂时本程序是如何使用这些技术的

@Repository  
public interface BasicInfoRepository extends PagingAndSortingRepository<BasicInfo, String> {  
 List<BasicInfo> findAll();  
}

然后对于实体类中

@Entity  
public class BasicInfo {  
  
 */\*\*  
 \* 股票编号  
 \*/* @Id  
 private String code;  
 }

以上只是代码的片段，但是的确十分方便，只需要简单的统一声明就能够实现相应的效果。

(4) Lombok

这又是一个简单的声明，替代了最为基础的get，set方法以及构造函数等等，他能够为我们节省大量的代码行数，同时也会为我们节约大量的时间，因为他的使用便利，我们在项目中大量的使用了这个声明，虽然只是最为基础的效果但是也为我们带来了便利。

@Data  
public class BasicInfo {

}

(5) RestTemplate

一个被宣称符合RESRful标准的网络访问工具，使用它的初衷是为替代现有的OKHttp或者其他的访问工具，将网络访问简化为最简单的配置过程，使用它的步骤也并不是十分繁琐，分成三部分，注册配置，声明，使用。

@Bean  
public RestTemplate restTemplate(ClientHttpRequestFactory factory){  
 return new RestTemplate(factory);  
}

@Resource  
RestTemplate restTemplate;

String resultStr = restTemplate.getForObject(String.*format*(urlHistory, symbolStr), String.class);

以上三个步骤使用restTemplate访问网络。

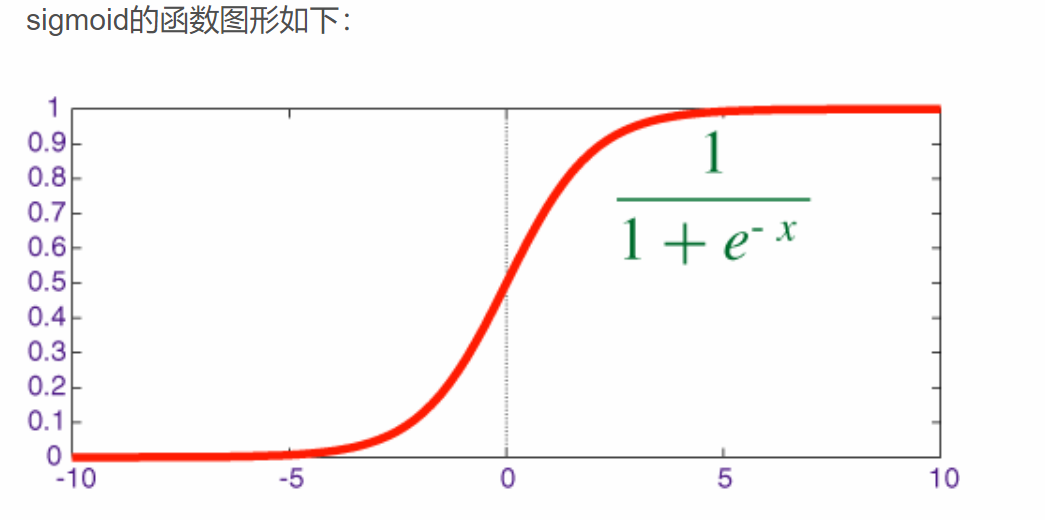
## 4.2 核心代码与算法

### 4.2.1核心代码理论基础

**基础知识------Logistic Regression（逻辑回归）**

首先先介绍一下**sigmoid函数**

这是一个很简单的公式，而他实现的主要内容也就是



他最重要的一个特性就是正一条曲线处于0-1之间，并且逐渐趋向于平缓。

**问题的转化**

我们在实际操作股票中往往有三种情况

1. 股票价格上升
2. 股票价格下降
3. 股票价格持平

对于股票投资者而言，我们其实只有两种状态，非赚即亏，使用我们能够将股票投资问题的结果简单的变为一个分类问题，并且这个分类问题是一个二分类问题。既然是一个二分类问题，我们就可以将它描述为最为简单的数学语言，0或1，0代表下降，1表示上升。

同时在股票操作中，我们会观察许多维度，构成本程序的基本理论就是，除去人为因素，股票价格被大量的维度所决定，而股票本身不决定自身价值。简而言之，也就是单一只股票不受到人为因素干扰时，决定股票价格走势的只有股票的参考维度。根据这个结论，我们能够对于每个独立的股票得到这样的公式，每一个维度，都分为不同权重决定了股票价格。通过数学化的表达也就是以下公式。其中D就是我们的结果。



此时问题的便转化为了，我们需要提供出一个公式或者方法，通过不同维度去评判这支股票，以及另一种思路，在大量的计算下股票是否能够呈现出一种概率分布，能够区分上升的股票以及下降的股票。

**准备阶段**

我们在上一阶段定义了问题的关键，而在准备阶段，我们需要定义什么是涨或者跌，我们要如何用数学语言告诉计算机一个显而易见的事实，我们需要将概念定义出来。

股票的涨幅：

这样的划分更加减少了外界对于股票价格影响，同时我们表明了一个观点，我们更加重视后三十日价格，明确我们的投资周期应该在两个月以上。

并且我们知道股票的各位维度都是不相同的，我们无法抱有绝对的态度说一个价值4000元一手的股票在统计学上一定有区别于1000一手的股票，或者说有其他的关系，但是我们无法判断，并且因为差异过大，无法作为一个有效的计算，使用我们的下一步就是将所有的维度降权为同一权重。

此时，**sigmoid函数将会被使用，我们将所有的值都将到-1至1之间，虽然我们知道由于函数的关系，我们无法将所有的权重全部消除，但是由于他们的值被无限地逼近，从原来的倍数级下降到了合理的范围之内，同时我们也保留了每个数值所包含的自身价值，而更加重要的是，我们的确对于股票与股票之间的差异逐渐抹去，而更加重要的是，我们也磨平了股票自身维度的权重对于自身的影响，我们将所有的维度降至一定的范围之内，40元的单价，不会再对每年0.07的净利润增长率造成过大的影响，这才是我们真正想要看到的，也就是所谓的消除外部权重影响。**

具体的算法其实在数据库设计的时候已经给出了，大家可以翻到那一页回顾一下内容，以及各个维度的处理方式。

**具体的数据处理过程**

由以上篇幅可以得出一个可以知道，我们的数据需要经过处理才能使用，于是我们将会在这里再次确认几个概念性词汇，原始数据，一级计算单元，二级计算单元。

这些概念有助于我们更加可分离性的处理数据，提供了程序解耦的条件，并且当系统运行失败的时候我们可以很好地保留一些。

原始数据：顾名思义，这是未经过处理的原生数据，直接的爬虫结果。

一级计算单元：经历过一定的数据整合与数据选取，我们在这一步率先的确定下我们之后想要使用的维度，但是这些数据还是符合了原始性，也就是没有经历过处理的。

二级计算单元：这一层是经过除权的最终参与计算的数据，我们通过一些计算公式将数据处理成了我们想要的样子，这里是对于以上公式的第一次运用。

举例：

经过这些数据的处理，我们才可以进行下一步的运算，而至此所有的数据准备工作结束。

**方法一：公式预测与回归**

既然我们希望得到一个公式，我们就需要大量的数据进行实验，这也就是我们实现了爬虫功能的目的，我们利用公式

很明显我们无法得出一个二分类的结果，只要数据量无限大时这个函数的结果呈现出的趋向于无限多种结果，使用我们将引入另一个概念，分类阙值（也成为判定阙值），逻辑回归返回的其实是一个类似于概率的东西，举一个简单的例子，一个垃圾邮件判断系统，当返回结果为0.6时，我们无法判断它究竟是不是垃圾邮件，而我们确立了阙值，我们通过计算我们认为高于0.5的全部都是垃圾邮件，这个时候我们会得到一个分类函数，当函数值大于某个值的时候我们将它分为一类，而小于的时候我们将他奉为另一类。如以下公式

既然确立了这个概念，我们需要大量的随机数来支持运算。在每一次运算之后我们必须进行回归，也就是我们会使用这个随机出来的公式对其他股票进行一样的操作，来进行一个调整，以此来不断的拟合回归，知道这个公式能够十分正确的体现出一个二类分类公式，我们估计这个操作应该有起码超过亿亿次的计算，而经过之后的不断试验，我们会发现性能瓶颈并不会存在于CPU计算能力上，而是内存读写速度上，在大量数据运算后，内存命中率会大幅度下降，并且因为大量的数据，甚至会有内存溢出的情况。

大致的过程可以描述为以下伪代码

Random random;

for(calculationUnit){

for(random){

for(Random x){

Double result = Random \* calculationUnit;

Int sumUp;

Int sumDown;

If(rate > x){

sumUp++;

}else{

sumDown;

}

}

}

}

我们举一个实际的例子，

运算复杂度为X^3

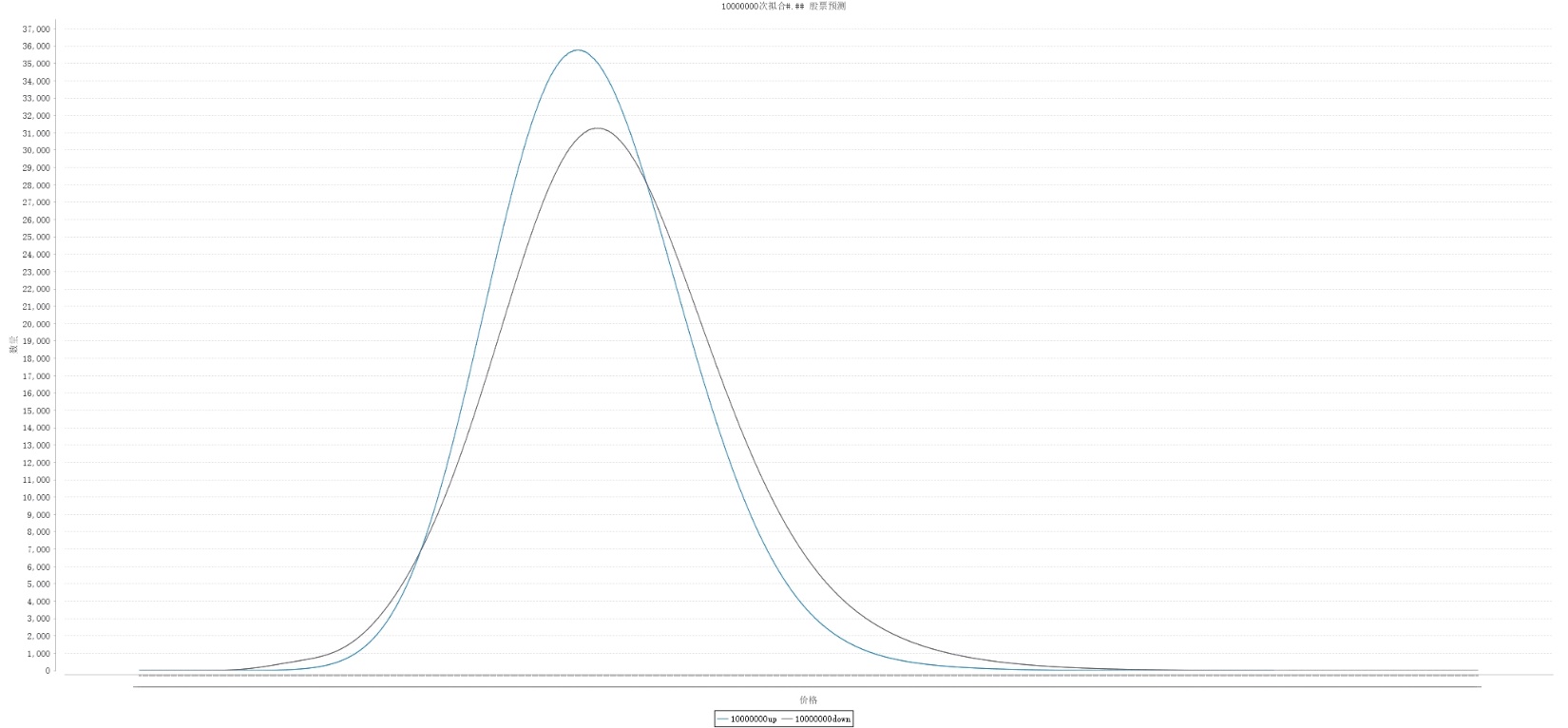
**方法二：概率分布**

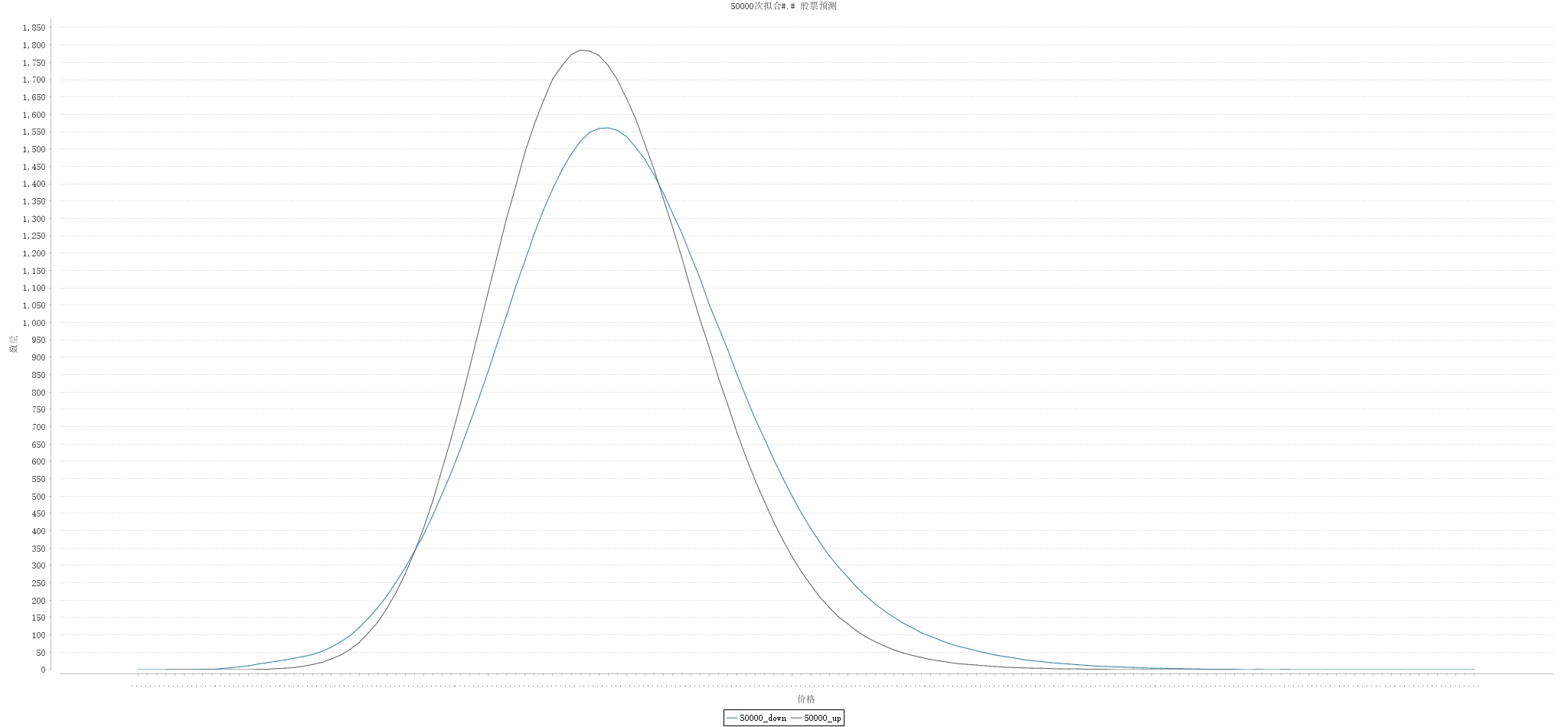
概率分布较之上述条件更加容易理解，因为逻辑S回归函数，但他进行运算的时候一定会呈现出一种概率分布，并且当数据量越大的时候概率分布的准确性将会越高，可以推断出一件事情，当一个函数符合概率分布时，将该函数以大量不同函数值相加，输出的值依然将会符合概率分布。



根据这个推论，我们可以猜测，当股票的各维度与一个符合概率分布的函数进行相称求和，他的输出一定也是一个符合概率分布的结果。如果符合概率分布，并且上涨与下降股票之间如果没有明显的概率分布区别的话，我们可以认为这个概率分布函数无法作为涨跌的依据，

但是结果并不是这样，涨跌两种股票呈现出了极为明显的概率分布差异。

10000000次拟合#.## 股票预测



50000次拟合#.# 股票预测

上图就很好的解释了我们的问题，很明显上涨或下降的股票会呈现出不同的概率分布。

# 5 系统部署与结果

## 5.1系统部署

需要在运行的服务器上安装mongo与MySQL并且改变springBoot环境配置文件，使用mvn clean install –Dmaven.test.skip=true 打包后运行Jar包即可，无过多操作。

## 5.2结果

共尝试两种计算方式。

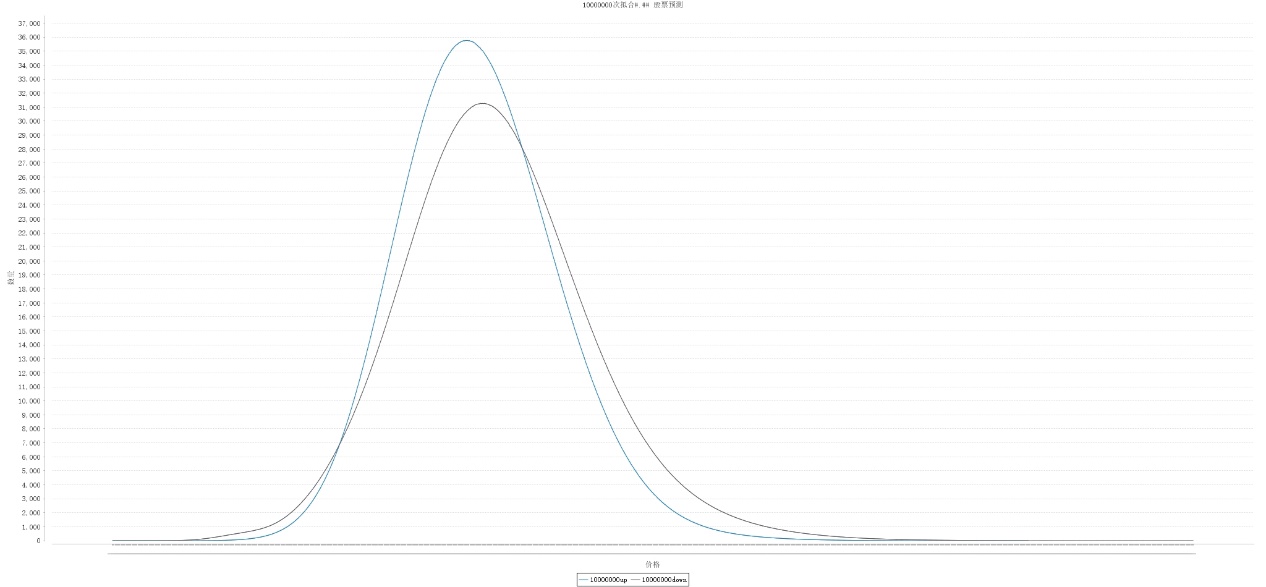
公式：

基本宣告失败

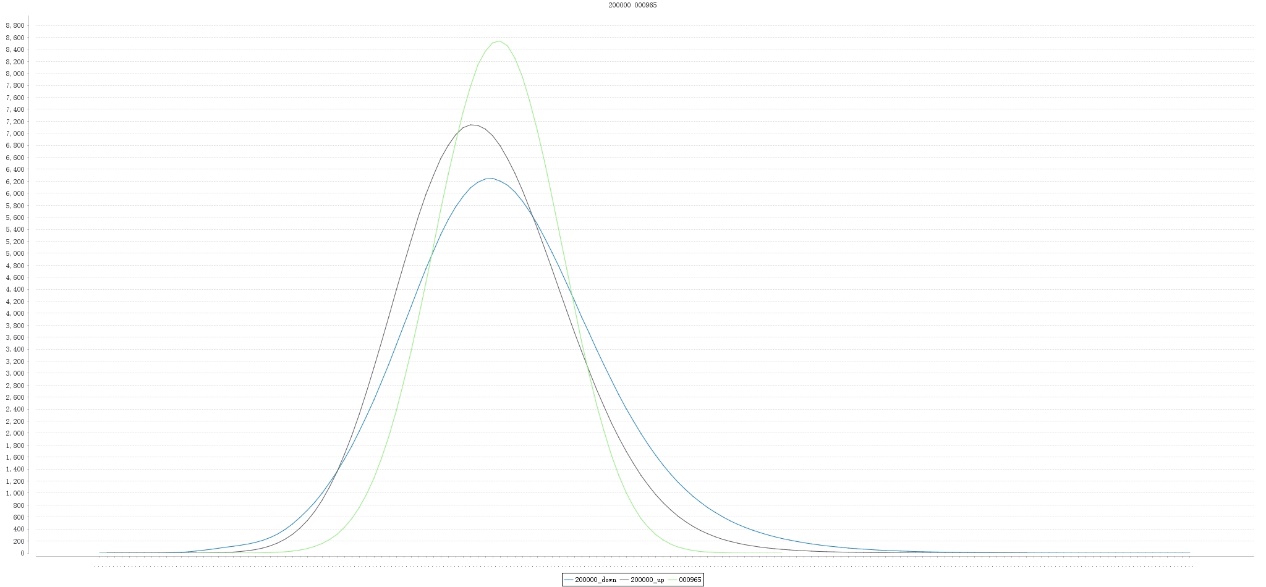


运算结果十分符合数学规律，找不到特征点。

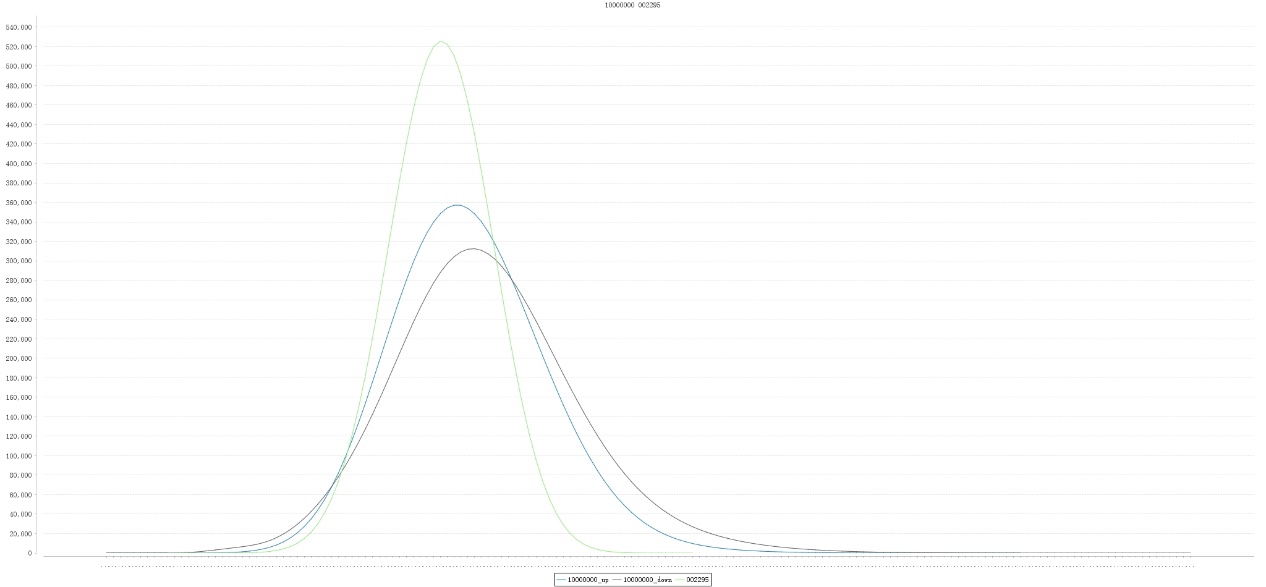
图标法：



拟合图



预测图该股票为下跌



预测图该股票为上涨

# 6.总结部分

系统能够大致是实现预测功能，但是并没有实现公式的计算，这十分可惜，因为我的代码是另外一种思路，从一开始就想要预测真正的涨跌数，而非真正的进行二分类，如果说什么系统地方需要改进的应该是这里。

在代码的后面一阶段，因为时间问题，慢慢开始没有使用特别标准的代码规范，虽然依然十分清晰，但是这个还需要改进。

同时这个项目的确十分的好玩，爬虫部分我花费了一个礼拜的时间就已经完成了，而当了预测模块，我花费了一个半月左右来完成，期间也被老师敦促过进度，那个时候依然在不停的思考中，每天闲暇时间总是在阅览股票知识，而后的一段时间中，我又学习神经网络的知识，每天都会抱着一本笔记本，不停的拿着钢笔敲击着纸头，不停的设计演算，希望能够找出最好的数学方法，虽然直至现在我依旧没有完全理解神经网络，但是不得不说，这些学习都给我提供了不同的视野。

同时也是因为这个毕业设计，我感觉到，思考是一个优秀的程序应做的事情，而不是简单的实现。

在这个项目中也有瓶颈，我的台式机在链表长度大于一千万以后将会有一个肉眼可见的性能下降，同样的列表，在九百万后会有一个明显的下降，而克服这些困难就显得尤为有趣。

而解决这些给了我充实感与趣味。

# 致谢

感谢父母，感谢各位老师，也十分感谢我的指导老师周平老师。

# 参考文献

[1]Google. Google Java Style Guide[OL]. <https://google.github.io/styleguide/javaguide.html>

[2]Scott Oaks.Java Performance The Definitive Guide[C].O’REILLY

[3]美团点评技术团队.[ OL]. Logistic Regression 模型简介

[4]Google.Machine Learning Crash Course[OL]. <https://developers.google.com/machine-learning/crash-course/>

[5]王福强.SpringBoot揭秘快速构建微服务体系[C].机械工业出版社