南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目 基于XGBoost和LSTM的智能监控系统的设计与实现

作 者 姓 名 张双江

学科、专业名称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇 教授

2019 年 5 月 27 日

学　　号：**MF1732172**

论文答辩日期：**2019 年 5 月 8 日**

指 导 教 师：　　（签字）

The Design and Implementation of Intelligent Monitoring System Based on XGBoost and LSTM

By

Shuangjiang Zhang

Supervised by

Professor **Zhenyu Chen**

A Thesis
Submitted to the Software Institute
and the Graduate School
of Nanjing University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

Software Institute

May 2019

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于XGBoost和LSTM的智能监控系统的设计与实现

工程硕士（软件工程领域） 专业 2017 级硕士生姓名： 张双江

指导教师（姓名、职称）： 陈振宇 教授

摘 要

软件质量已经影响到人们的日常工作和生活。监控系统是软件质量保障的其中一种手段。当前监控系统大多采用固定阈值报警的方式检测异常，由于监控指标之间的关联性及不同服务器的差异性，采用固定阈值方式检测异常容易产生大量误报和漏报。监控系统的趋势预测功能还不成熟，运维人员不能及早发现系统问题和资源瓶颈，不能及时调度资源以保证系统稳定运行。因此，采用机器学习技术进一步完善监控系统对于软件质量保障具有重要的价值。

本文结合工业场景分析系统需求，设计与实现了一个基于XGBoost和LSTM的智能监控系统。XGBoost适合多维数据二分类问题，能够挖掘监控数据间的关联。LSTM适合长时间序列趋势预测，能够对具有时间特征的监控数据进行准确的趋势预测。本系统引入孤立森林算法完成数据标注预处理工作，从而减少运维人员数据标注工作量，进一步通过迭代更新数据标签并重置模型的方式不断提高异常检测和趋势预测准确率。本系统主要分为四个模块。监控模块负责监控数据的采集和存储，当系统出现异常数据时能通过微信企业号和邮件通知相关用户。数据处理模块负责数据的处理和管理，为异常检测模块和趋势预测模块提供服务。异常检测模块提供实时数据异常检测功能，接收实时数据后调用相应模型进行异常检测。趋势预测模块提供数据未来趋势预测功能，能够预测数据未来趋势并通过可视化形式展示。本系统采用进程池后台异步训练模型保证系统响应速度，采用Redis缓存数据库提高系统性能和读写速度。

本系统目前已经部署试用。经过30天的应用效果评估，结果表明本系统异常检测准确率达到86.57%，趋势预测值与真实值的均方根误差小。和传统的固定阈值报警方式相比，本系统提升了27.53%的报警准确率，减少了39.13%的漏报。综上所述，本系统能够提供较为准确的异常检测和趋势预测功能，有效减少误报和漏报，减轻运维人员设置海量监控阈值工作量。

关键词： 异常检测，趋势预测，智能监控系统，XGBoost，LSTM

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of Intelligent Monitoring System Based on XGBoost and LSTM

SPECIALIZATION: Software Engineering

POSTGRADUATE: Shuangjiang Zhang

MENTOR: Professor Zhenyu Chen

Abstract

Software quality has affected people's daily work and life. The monitoring system is one of the means of software quality assurance. At present, most of the current monitoring systems adopt fixed threshold alarms method to detect anomalies. Due to the correlation between monitoring indicators and the differences of different servers, it is easy to generate a large number of false positives and false negatives when using fixed threshold method to detect abnormalities. The trend prediction function of the monitoring system is still immature. Operation and maintenance personnel cannot identify system problems and resource bottlenecks early to ensure stable operation of the system by adjusting resources. Therefore, using machine learning technology to further improve the monitoring system is important for software quality assurance.

An intelligent monitoring system based on XGBoost and LSTM in combination with the requirements of industrial scene has been designed and implemented, which is presented in this thesis. XGBoost is suitable for multi-dimensional data classification and it can mine the correlation between monitoring data. LSTM is suitable for long-term sequence trend prediction and it can accurate trend prediction of monitoring data with time characteristics. The system introduces the Isolation Forest algorithm to implement data annotation preprocessing, thereby reducing the workload of data labeling for operation and maintenance personnel. Furthermore, the accuracy of anomaly detection and trend prediction is improved by iteratively updating data labels and resetting models. The system is mainly divided into four modules. The monitoring module is responsible for the collection and storage of monitoring data. When abnormal data appears in the system, the relevant users can be notified through WeChat and email. The data processing module is mainly responsible for data processing and management, and

provides services for the anomaly detection module and the trend prediction module. The anomaly detection module provides the function of real-time data anomaly detection. After receiving real-time data, the corresponding model is called for anomaly detection. The trend prediction module provides the function of data future trend prediction, which can predict future trends data and visualize them. The system uses the process pool background asynchronous training model to ensure the system response speed and uses the Redis cache database to increase the speed of reading and writing.

At present, the system has been deployed for trial use. After 30 days of application evaluation, the results show that the accuracy of the anomaly detection of this system reaches 86.57%, and the root mean square error of the trend prediction value and the true value is small. Compared with the traditional fixed threshold alarm method, the system improves the alarm accuracy rate by 27.53% and reduces the false negative report by 39.13%. In summary, the system can provide accurate anomaly detection and trend prediction functions, effectively reducing false positives and false negatives, and reducing the workload of the operation and maintenance personnel to set a large number of monitoring thresholds.

Keywords: Anomaly Detection, Trend Prediction, Intelligent Monitoring System, XGBoost, LSTM

目　录

表　目　录

图　目　录

第一章　引言

1.1　项目背景和意义

随着互联网的高速发展，人们的生活越来越依赖各种软件系统。一个企业的软件系统的正常运行不仅关系到用户使用体验，也关系到一个企业的名誉和利益，这使得能保障业务可靠和稳定运行的监控系统越来越重要。但是随着软件系统业务的扩展，海量的监控数据、错综复杂的监控指标和大量监控指标阈值的设定大大增加了运维人员的工作压力。目前常用的监控系统大多是通过设置固定阈值的方式实现异常报警，即当监控项数据超过设定的阈值时会触发报警，这种方式存在三个弊端：一是单监控项阈值设定，忽略了监控项之间的关联性；二是不同环境下监控项阈值不一样，难以设置准确的报警阈值；三是海量监控项阈值设定耗时耗力、工作量巨大。前两点造成了异常报警不准确，第三点增加了运维人员的工作量。另外，对软件系统使用资源的预测也十分重要，运维人员如果不能预测数据未来趋势变化，提前对资源进行调度，当系统出现资源瓶颈和故障时，难以在短时间内解决问题。趋势预测功能可以准确预测到资源瓶颈，运维人员可以提前发现问题并进行资源调度，在保证系统稳定性和可用性上具有十分重要的作用。异常检测和趋势预测是目前监控产品缺少但是又能极大提高运维效率的两个功能，这时候设计一个具有自动检测监控项异常和对监控项指标进行趋势预测功能的监控系统就变得尤其重要。

本论文旨在设计一款利用XGBoost和LSTM算法模型对海量监控项进行异常检测和趋势预测的监控系统。该系统以开源监控软件Zabbix为基础，使用XGBoost训练异常检测模型，通过分析系统历史监控项数据进而准确地对实时监控数据进行异常检测，从而减少运维人员设置海量监控项阈值的工作量，提高异常报警准确率，减少误报和漏报。系统通过LSTM算法预测监控数据未来趋势变化，运维人员可以查看服务器资源未来趋势走向，提前对服务器资源扩缩容，从而保证系统稳定运行，保障用户体验。运维人员可以针对服务器集群中不同服务器的监控项组合训练不同的异常检测和趋势预测模型模型，最终形成互不影响、功能各异的模型群，不同的模型适用于不同的服务器数据，从而减少模型之间耦合。服务器监控项发生变化时，系统只需要重新训练该监控项对应的模型，所以系统能够有效的应对监控项增减和变更的情况。模型在使用的过程中，运维人员可以对异常检测模型判断不准确的数据重新进行数据标注，然后重置模型，通过迭代的方式不断提高模型的准确率。

1.2　研究现状

本系统是对实时监控系统数据进行异常检测与趋势预测，所以本节详细介绍实时监控系统应用现状、异常检测算法研究现状和趋势预测算法研究现状。

1.2.1　实时监控系统应用现状

目前常见的监控产品主要有阿里云监控、360网站服务监控、监控宝、Nagios、Ganglia 和Zabbix等，本节详细介绍上述产品的主要功能及优缺点。

阿里云监控与阿里云服务器天然集成，可用于监控站点和服务器信息，提供包括短信、邮件和旺旺多种告警方式，但是只能用于阿里云服务器，无法监控企业私有服务器。360 网站服务监控支持HTTP、PING、域名DNS和服务器监控，可用于监控网站和服务器，但是监控项类型少且数据仅能保存15天。监控宝提供网站监控、页面性能监控、API监控、基础设施监控和服务监控等功能，提供多种消息通知方式，具有部署方便和可定制监控内容等优势。

Nagios是一款监视系统运行状态和网络信息的监控系统，侧重于监控服务的可用性[1]，可以监控网络服务（SMTP、POP3、HTTP 和PING等）和服务器资源（CPU使用率、磁盘利用率、内存使用率）。Nagios可以在Linux和Unix系统运行，提供Web页面展示监控信息和异常告警通知功能 [2]，但是Nagios扩展困难且无法查看历史数据定位故障原因。

Ganglia是由加州大学伯克利分校发起的开源集群监控项目，其核心组件包含gmond、gmetad和Web前端[3]。Ganglia系统服务端和客户端之间的连接开销非常低，适用于监控大规模集群系统的性能，例如CPU使用率、内存使用率、硬盘利用率和网络流量情况等。Ganglia系统提供可视化界面，具有配置简单、部署方便、扩展性强和跨平台等优点，其缺点是缺乏消息通知机制，系统出现故障时不能及时通知相关人员。

Zabbix是一款应用广泛的企业级分布式监控系统 [4]，提供多种采集数据方式（主动和被动），并可以永久存储数据。Zabbix具有分布式架构，可用于监视网络、服务器和应用等状态，支持MySQL、PostgreSQL和SQLite等多种数据库存储数据，支持IPMI、JMX和SNMP等多种协议 [5]，支持多种语言，提供多种报警机制，支持用户自定义报警条件 [6]，提供灵活的API接口，支持用户编写脚本，支持自动发现服务器和网络设备。Zabbix具有安装部署简单、可二次开发、扩展性强、易于维护和管理、可跨平台使用和监控数据详尽等优点，其缺点是自定义监控项阈值报警设置繁琐，二次开发难度大。

1.2.2　异常检测算法研究现状

异常点检测（又称为离群点检测）是用于找出与预期差距很大的对象，这些与预期差距很大的对象被称为异常点或者离群点[7]。异常点检测最初应用于统计学领域，Knorr等人将其延伸到数据挖掘领域中 [8]，目前在生活中应用的十分广泛，例如信用卡欺诈检测，工业产品合格检测，图像检测等。异常点检测主要分为基于统计、最近邻、聚类和分类四类异常检测方法。

基于统计的异常检测方法通常会先假设一个概率模型或概率分布，并对数据进行拟合，观察实际数据的分布是否符合概率模型或概率分布，如果符合则认为是正常点，不符合则认为是异常点。常见的基于统计的异常点检测方法有高斯模型、回归模型和核密度函数法等 [9]。基于统计的异常检测方法以数学理论为基础，具有高效和高准确率的优点，但是也存在许多弊端，例如大多数分布模型只适用于单变量数据，无法分析高维数据。另外，基于统计的方法需要提前理解数学原理和数据分布特征，现实情况中很难做到这一点。

基于最近邻的异常检测方法比基于统计的方法更一般和普适，其基本假设是正常点都集中在密集的领域，异常点通常会远离其他点，该方法有两种计算方式，分别是计算样本K邻近点的距离和计算样本区域和领域的密度。常见的基于距离的方法有KNN（K-nearest neighbor）[10]，KNN基本思想是一个对象的异常点得分由该点到离该点最近的K个点的距离决定，常用的距离有曼哈顿和欧几里得距离等 [11]。基于密度的方法是对基于距离方法的进一步扩展，常见的基于密度的算法有局部异常因子方法LOF [12]和连通异常因子方法COF [13]等。Breunig等人在2000年提出局部异常因子LOF的方法 [14]，通过计算数据局部密度和领域密度之比来确定局部异常因子，局部异常因子较大的数据更可能是异常点。Papadimitriou等人在2003年提出使用异常点的偏离程度来判断异常 [15]。基于最近邻的异常检测算法的优点是无监督且不需要假设分布模型，缺点是算法复杂度大，处理大量数据时速度慢，不适合实时数据的异常检测。

基于聚类的异常检测方法属于无监督算法 [16]。聚类可以分为三种情况：第一种是假设正常点归属某一个簇，而异常点不属于任何一个簇，例如Ester等人提出的DBSCAN算法 [17]；第二种是假设正常样本离聚类中心较近，而异常样本离聚类中心较远，例如K-means算法 [18]；第三种是假设正常点归属于数量多且密集的簇，而异常点归属于数量少且稀疏的簇，例如He等人提出的CBLOF算法 [19]。

基于分类的异常检测方法使用有标注的数据训练模型，一般分为训练和测试两个阶段：训练阶段对大量有标注的数据集进行分类得到模型；测试阶段使用模型对测试数据进行分类判断是否异常。常见的分类算法有支持向量机

（SVM）、决策树、logistic回归等。基于分类的异常检测方法的优点是可用于大规模高维数据的异常值检测，分类方法容易学习掌握。

1.2.3　趋势预测算法研究现状

趋势预测大多是以时间为自变量，根据某一对象在过去一段时间内的表现预测未来一段时间内该对象可能的趋势。实时监控系统中要处理的数据往往都具有时间特征，具有时间特征的数据序列被称为时间序列。时间序列趋势预测就是根据时间序列所反映出来的发展过程、方向和趋势进行扩展和延伸，以预测未来一段时间可能出现的状况。预测时首先分析已有时间序列，从中找出随时间变化的规律并得出一定模式，最后用这个模式来预测将来的情况 [20]，例如预测网站的请求量和用户增长趋势等。

时间序列分为平稳序列和非平稳序列：平稳性是指一个时间序列的均值、方差和周期性都不发生变化，平稳序列预测方法主要有移动平均法和指数平滑法等 [21]；非平稳时间序列一般具有比较明显的特征，如趋势性、周期性和随机性，常见的趋势预测算法有ARIMA自回归移动平均模型、贝叶斯模型、神经网络算法等 [22]。Contreras在2003年使用ARIMA预测电费趋势 [23]。郑为中等人在2005年使用贝叶斯模型预测交通情况 [24]。Connor等人在1994年将循环神经网络（RNN）应用于时间序列趋势预测 [25]。Hochreiter等人在1997年提出长短期记忆神经网络（LSTM）[26]，LSTM与RNN的区别在于LSTM算法中增加了记忆单元，可以控制历史信息记忆的程度。Shi等人在2015年利用LSTM预测降雨量 [27]。Alahi等人在2016年使用LSTM预测人体轨迹情况 [28]。Sezer等人在2018年基于二维卷积神经网络提出CNN-TA算法用于金融交易 [29]。Wang等人在2019年提出利用稀疏表示和模糊集理论对股票进行趋势预测 [30]。

1.3　本文主要研究工作

为解决目前常用监控产品异常检测不准确和缺乏趋势预测功能的问题，本文设计和实现了基于XGBoost和LSTM的智能监控系统。该系统提供智能监控服务，提供准确的实时数据异常检测和趋势预测功能，从而减轻运维人员工作量，提高工作效率。本文主要研究工作如下。

本文首先分析目前常用监控产品的功能和优缺点，选取Zabbix用于采集监控数据。然后分析常用的异常检测和趋势预测算法，选取XGBoost作为异常检测算法，LSTM作为趋势预测算法，孤立森林作为数据标注预处理算法开发智能监控系统。最后分析智能监控系统用户需求，根据需求将系统分为监控模块、数据处理模块、异常检测模块和趋势预测模块。

本文设计并实现了监控模块，该模块可以采集被监控服务器的信息和应用信息并存储（例如CPU使用率、内存使用率、网络带宽、网站运行状态和MySQL服务状态等），并通过Python脚本实现微信和邮件消息通知功能。

本文设计并实现了数据处理模块，该模块实现了批量数据标注、查看异常检测模型信息、查看趋势预测模型信息、查看数据集信息、使用孤立森林算法完成数据标注预处理、将模型持久化至磁盘和Redis等功能。该模块接收用户请求并对数据进行处理和管理，为异常检测和趋势预测模块提供服务。

本文设计并实现了异常检测模块，首先通过实验确定XGBoost模型最优参数，然后实现了训练和重置异常检测模型。用户可以根据数据集训练多个不同的异常检测模型，当被监控系统产生实时数据时，异常检测模块根据实时数据来源加载对应的异常检测模型检测异常并通过监控模块报警。

本文设计并实现了趋势预测模块，该模块以LSTM算法为基础，提供训练和重置多个不同的趋势预测模型、预测不同数据的未来趋势走向的功能。

本文提供风格美观、交互友好的前端界面，用户可以通过界面上传数据集、批量进行数据标注、训练和重置模型、查看模型和数据信息、查看异常列表和对数据进行趋势预测等。

1.4　本文组织结构

第一章是引言，主要介绍了当前常用监控产品面临的问题，阐述了当前常用监控产品的功能及优缺点，介绍了异常检测算法和趋势预测算法的研究现状，最后说明了本文主要工作。

第二章是相关技术综述，主要介绍了Zabbix监控基本功能和组织架构，Django框架和Redis缓存功能和优点，孤立森林、XGBoost和LSTM算法的原理及优势，并分别解释了本系统使用上述技术、算法和工具的原因。

第三章是智能监控系统的需求分析与设计，首先介绍了系统功能性和非功能性需求，然后详细介绍了系统总体架构，最后介绍了监控模块、数据处理模块、异常检测模块和趋势预测模块的架构、类图、流程和数据库设计。

第四章是智能监控系统的实现和测试，首先介绍了四个功能模块的顺序图和代码实现，通过实验完成XGBoost算法模型调参过程，然后对系统进行功能测试和性能测试，并展现系统运行效果，最后介绍了系统效果评估结果，得出本系统异常报警和趋势预测的效果。

第五章是总结与展望，对本系统和论文进行工作总结，指出下一步工作方向，为今后研究提出改进意见。

第二章　相关技术综述

本章在进行系统分析设计与实现之前，详细介绍与系统设计实现相关的技术理论，完成了对系统的技术选型。系统选取Zabbix作为监控系统的基础支撑，Django作为开发框架，Redis作为缓存数据库，孤立森林作为数据标注预处理算法，XGBoost作为异常检测的算法，LSTM算法作为趋势预测的算法。本章将详细介绍Zabbix、Django、Redis、孤立森林、XGBoost和LSTM。

2.1　Zabbix监控系统

2.1.1　Zabbix组件介绍

Zabbix是一款由Alexei Vladishev开发的应用广泛、性能优良、功能强大的企业级开源监控软件[31]。Zabbix系统由多个组件组成，分别是Zabbix Server（Zabbix服务器），可选组件Zabbix Agent（被监控设备），可选组件Zabbix Proxy（Zabbix代理服务器），Zabbix Web Frontend（Web前端组件）和数据库。

Zabbix Server是核心组件，用于收集Zabbix Agent和SNMP等发送的数据并写入数据库。所有的配置、数据、用户操作和消息通知等都由它组织进行。Zabbix Agent是可选组件，部署在被监控设备上，用于采集该设备本地数据，例如CPU使用率、内存使用率、磁盘使用率、网络带宽和自定义数据等，然后将数据发送给Zabbix Server端或者Zabbix Proxy端。Zabbix支持多种采集方式，Zabbix Agent只是众多采集方式中的一种，所以它并不是必要组件。Zabbix Proxy是可选组件，常用于分布式环境中，用于减轻Zabbix Server端的数据采集压力，当有大量Zabbix Agent端数据需要采集时，仅由一个Zabbix Server处理大量数据压力巨大，此时可采用多个Zabbix Proxy代理服务器接收Agent端发送的数据，整理后发送至Server端以减轻Server端压力。Zabbix Web Frontend用于可视化展示采集数据、报警情况和用户信息等，可以采用多种图表展示数据。

2.1.2　Zabbix架构介绍

Zabbix结构分为通用架构（Client/Server）和分布式架构（Client/Node/Server或者Client/Proxy/Server）两类。通用架构Client/Server中，Zabbix Client采集数据并直接发送给Zabbix Server处理，适用于小型规模的服务器群。分布式架构中Node与Proxy区别在于Node节点能够存储节点自身的配置和数据，而Proxy节

点配置和数据都储存在Server端，采用代理的方式可以减轻Zabbix Server端的处理压力，适用于大规模服务器集群监控。

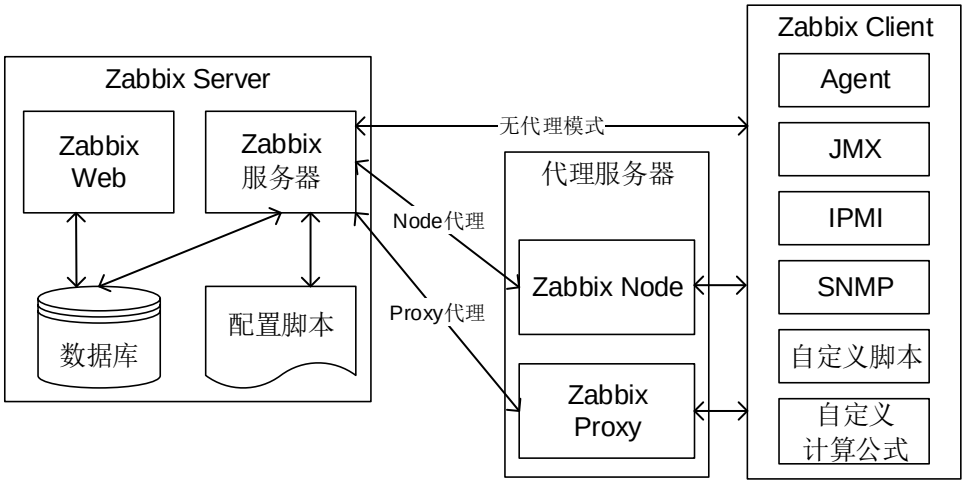

图 2.1: Zabbix架构图

Zabbix架构图如图2.1所示。如果要监控单机或者小规模的服务器集群，可以选用Zabbix Server与Zabbix Client直接相连的Client/Server架构，如果要监控大规模的服务器集群，可以选用Client/Node/Server或者Client/Proxy/Server分布式架构。Zabbix可以采用IPMI（智能型平台管理接口）、SNMP（简单网络管理协议）、JMX（Java 管理扩展）和Agent等多种方式采集数据并配置采集频率和采集规则。Zabbix可以采集通用信息，例如CPU使用率和磁盘使用率等，也可以采集自定义信息，例如API接口调用次数、网站在线人数、网站日活量、Docker容器占用资源和运行情况等信息。

2.1.3　Zabbix的优势

Zabbix提供多种数据采集方式和大量可二次开发的API接口，具有强大的告警机制和用户自定义告警条件功能、界面展示等功能，非常适合服务器监控。Zabbix支持跨平台和分布式部署，支持自动发现服务器和网络设备，具有易于维护和管理、扩展性强、安装部署简单便捷、监控数据详尽等优点。

本系统充分利用Zabbix监控项详尽、部署快捷简单、扩展性强、可二次开发等优点，采用Zabbix V3.4版本作为监控系统基础，采用Server/Client架构方式部署，使用Agent、ICMP和自定义脚本等多种方式采集详尽的数据，使用自定义Python脚本实现微信和邮件报警功能。数据是系统所有功能的基础，所以Zabbix必须及时采集详尽准确的数据。

2.2　Django框架

2.2.1　Django框架介绍

Django是一款由Python开发，采用MVT（Model-View-Template）设计模式的Web应用框架[32]，其核心包括对象关系映射器（ORM）、基于正则表达式的URL分发器、视图系统和模板系统。

Django支持原生SQL语句和对象关系映射器操作数据库。对象关系映射器可以将数据模型与数据库表关联起来，从而避免编写繁琐的SQL语句。基于正则表达式的URL分发器可以通过匹配URL，将用户访问请求路径映射到View中对应的函数里并处理请求。视图系统提供一个集中存放视图函数的文件View.py，主要负责接收URL请求，处理请求并通过HttpResponse对象返回结果给用户。模板系统主要决定前端如何组织和展示数据，Django提供多种标签形式展示后端数据，例如单变量标签{{value}}、块标签{%block%}和循环标签{%for%}等[33]。

2.2.2　Django的优势

本系统涉及多种机器学习算法和数据处理，所以首选Python作为开发语言，Django与Python天然集成，具有灵活的URL映射、丰富的模板系统、强大的数据库访问组件ORM、完善的后台管理功能、完善的错误提示信息、大量常用的工具和框架等优势。

本系统采用Django作为开发框架，充分利用其丰富的模板、URL映射和数据库映射ORM等功能，配合Python快速开发。前端利用其模板结合Bootstrap组件开发风格统一、交互友好的界面。

2.3　Redis缓存数据库

2.3.1　Redis数据库介绍

Redis是一个遵守BSD协议的开源高性能非关系型内存数据库，可以用于存储多种数据类型，例如列表、集合、有序集合、哈希表和字符串等 [34]。Redis支持数据持久化，可以将内存中的数据保存至磁盘，重启时可以将磁盘数据加载至内存，Redis支持数据备份，可以使用主从模型备份。由于其性能高、读取速度快、提供持久化功能、支持原子性操作和可配置保存时间等优点，Redis作为内存数据库应用广泛。

2.3.2 Redis的优势

Redis性能高，读写速度快，其读速度是110000次/s，写速度是81000次/s。Redis的所有操作具有原子性，一个事务（transaction）中的所有操作，只会全部完成或者全部不完成，不会在中间环节结束。Redis功能丰富，可以设置缓存过期时间，支持publish/subscribe功能，支持持久化，可以自定义持久化规则将内存中数据持久化到磁盘。

因为系统可能存有大量模型对象，将所有模型对象存放在内存会造成大量内存浪费。本系统采用Redis作为数据库缓存，配合磁盘持久化模型对象，避免将所有模型存放在内存导致浪费内存，从而提高模型对象读取速度和系统性能。当模型创建成功或者被使用时，系统将模型对象存储至Redis并设置过期时间为2小时，如果2小时内没有使用该模型，则从Redis中清除。当系统需要再次使用该模型时，则从磁盘中读取模型对象重复上述步骤。

2.4 孤立森林算法

孤立森林（Isolation Forest）是由刘飞等人在2008年提出异常检测方法 [35]，是一种适用于连续数据的无监督异常检测方法。它不依赖于任何距离或密度测量，从而消除了距离和密度方法的计算成本。在孤立森林中，异常被定义为“容易被孤立的离群点”，即分布稀疏且离密度高的群体较远的点。分布稀疏的区域可以认为发生在该区域的概率较低，因而认为这些区域内的数据是异常的。孤立森林使用时具有线性的时间复杂度、恒定的训练时间和空间复杂度，其扩展能力强，能够快速处理大量数据和多维数据问题，在工业界应用广泛。

2.4.1 孤立森林原理介绍

孤立森林是一种无监督异常检测算法。理解孤立森林（以下简称iForest）前需要先理解孤立树（以下简称iTree）。iTree是一种随机二叉树，每个节点有两个子节点（左子树和右子树）或者没有节点（本身是叶子节点）。一个具有多维数据的数据集D在构建iTree时，首先随机选择数据的一个属性A和该属性的一个值V，然后根据属性A对数据集中每条数据进行分类，将A值小于V的数据放在左子树上，将A值大于V的数据放在右子树上。最后分别在左子树和右子树按上述过程递归构造，直到树的高度达到阈值或者数据集只剩一条或多条一样的数据时停止构造。

iTree构造完成后可以对数据进行预测，数据从iTree根节点开始搜索，直到该数据落到叶子节点上。假设从根节点到叶子节点的路径长度为$h(x)$，由于异

常值比较稀疏，在iTree中很快会被分到叶子节点上，所以其$h(x)$会很短。因此可以用$h(x)$来判断某数据是否为异常值，判断前对路径长度$h(x)$进行归一化处理，使所有样本指标处于同一数量级，方便综合评价和比较。一个包含n个样本的数据集，树的平均路径长度$c(n)$为：

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{2.1}$$

其中$H(i)$为调和数，该值可以估计为$ln(i)$+0.5772156(欧拉-马斯刻若尼常数)。$c(n)$为给定样本个数后的平均路径长度，用来归一化样本的路径长度。$E(h(x))$是样本x在所有iTree的平均高度值。样本x的异常分数定义为:

$$s(x,n) = 2^{\frac{-E(h(x))}{c(n)}} \tag{2.2}$$

由式(2.2)可以看出，当$E(h(x))$趋近于$c(n)$时，异常得分s值趋近于0.5，此时不能确定是否是异常，因为样本x的平均路径长度和树的路径平均长度相近。当$E(h(x))$趋近于0时，异常得分s值趋近于1，此时样本x 的平均路径长度较短，判定为异常点。当$E(h(x))$趋近于$n-1$时，异常得分s值趋近于0，此时样本x的平均路径长度接近$n-1$，判定为正常。

由于随机树是随机选择属性和属性值，所以最终得到的随机树是不稳定的，具有很大的随机性。但是如果结合多棵iTree形成iForest结果就会更准确和稳定。构建iForest时随机采样一部分数据集来构建一个iTree，保证不同iTree具有差异性。另外，如果数据集过大，异常值较多形成密集的区域会被判定为正常值，所以在每棵树构建时需要限制随机采样数据集的大小。iForest构建完成之后，使用式(2.2)计算样本的异常分数。

2.4.2　孤立森林的优势

孤立森林是一种无监督异常检测算法，不需要对数据集进行数据标注。其具有线性时间复杂度，训练和使用模型时对时间和空间要求低，只需要少量的数据集就能训练出准确率较高的模型。并且孤立森林可以通过增大构造iTree的数量提高模型准确率，可以使用分布式计算加快构建速度。孤立森立算法原理简单，容易理解和实现，适用于多维数据的无监督异常检测。

由于孤立森林无监督、数据处理速度快并且效果好。本系统采用孤立森林作为数据标注预处理算法，用于监控模块采集数据后进行初步数据标注，为数据异常检测提供数据基础。使用孤立森林算法进行数据标注极大地减少了运维人员数据标注的工作量，

2.5　XGBoost算法

XGBoost（极端梯度上升，Extreme Gradient Boosting）是陈天奇在2016 年提出的基于弱学习器集成的算法 [36]，弱学习器是指泛化性能比随机猜测高的学习器，例如对于二分类问题，如果精度略高于50%的分类器就可以认为是弱分类器。XGBoost允许选择多种弱学习器集成预测，将多种弱学习器的预测结果集成起来作为最终结果来提高准确率 [37]。XGBoost中可供选择的基学习器有决策树、逻辑回归和SVM等算法。Mitchell等人在2017年利用GPU提高XGBoost的计算速度 [38]，Fitriah等人在2017年利用数据降维提高XGBoost预测准确性 [39]，Ren等人在2017年利用XGBoost与神经网络进行图像分类 [40]。Zhang等人在2018年利用XGBoost检测风力发电机故障 [41]。由于XGBoost训练速度快并且效果好，常被应用到各大算法大赛中。

2.5.1　XGBoost原理介绍

XGBoost由很多CART（Classification And Regression Tree，分类回归树）集成 [42]。数据挖掘和机器学习中的决策树主要分为两种：一种是分类树，用于预测结果是否是归于某一类；另一种是回归树，用于预测结果具体的值。CART是分类树和回归树的总称。分类的目标是根据已知样本的特征判断样本属于何种已知类型，其结果是离散值。回归问题的目标是预测具体的值作为结果。分类和回归问题很相似，其本质都是根据样本特征（feature）和标签（label）得到某种映射关系。

XGBoost算法核心思想是每次迭代增加一棵树，用于拟合上次迭代过程预测值和真实值的残差，从而不断接近实际值。每个样本的得分就是该样本在每棵树中得分之和。训练模型相当于获得最佳参数θ，使训练数据x_i和标签y_i映射效果达到最优。在训练模型时需要定义目标函数来度量模型与训练数据的匹配程度，进而评估模型效果，目标函数一般由误差函数项L和模型复杂度函数Ω组成，如式(2.3)所示，本文选用均方根误差作为误差函数项$L(\theta)$，如式(2.4)所示。

$$Obj(\theta) = L(\theta) + \Omega(\theta) \tag{2.3}$$

$$L(\theta) = \sum_{i}(y_i - \hat{y}_j)^2 \tag{2.4}$$

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{2.5}$$

式(2.5)中K是树的数量，f是特征空间F中的一个特征，y_i是真实值，$\hat{y}_i$是根据所有树得出的预测估计值，$L(\theta)$是所有数据预测值与真实值的均方根误差之和。在模型学习的过程中，使用加法策略，每次迭代新增一棵树用于拟合上一

次迭代得到的结果和真实值的残差，对上一次得到的结果进行修复。假设在第t步预测的值为$\hat{y}_i$，则每一步得到的预测值分别是:

$$\hat{y}_i^{(0)} = 0 \tag{2.6}$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \tag{2.7}$$

$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \tag{2.8}$$

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{2.9}$$

第t步的估计值$\hat{y}_i^{(t)}$是第$t-1$步估计值$\hat{y}_i^{(t-1)}$加上第t步的预测值$f_t(x_i)$。将结果带入误差函数项L中，并将误差函数的泰勒展开式提升到二阶。此时目标函数如式(2.10)所示，其中$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$，$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$，$\Omega(f)$是自定义模型复杂度函数。

$$\begin{aligned} Obj^{(t)} &= \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{n} \Omega(f_t) \\ &= \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \Omega(f_t) + constant \end{aligned} \tag{2.10}$$

XGBoost的复杂度包含两个部分：一部分是树中叶子节点的个数T，如果叶子非常多会造成过拟合；另一部分是树上叶子节点权重w_j的平方。所以模型复杂度函数$\Omega(f)$如式(2.11)所示。

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{2.11}$$

目标函数加入模型复杂度函数并去掉常数项后如式(2.12)所示，由于同一叶子节点上所有数据点的权重值相同，所以可以将所有数据点的求和转变成对叶子节点的求和，其中$G_j = \sum_{i \in I_j} g_i$，$H_j = \sum_{i \in I_j} h_i$，$I_j$表示所有的叶子节点。

$$\begin{aligned} Obj^{(t)} &= \sum_{i=1}^{n} [g_i w_q(x_i) + \frac{1}{2} h_i w_q(x_i)^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} [G_j w_j + \frac{1}{2}(H_i + \lambda) w_j^2] + \gamma T \end{aligned} \tag{2.12}$$

假设$w_j^* = -\frac{G_j}{H_j + \lambda}$，此时目标函数为$Obj^* = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$。目标函数越小时模型效果越好。XGBoost采用“贪心法”对决策树进行分割计算增益，每次对已有叶子节点进行分割并获取最大增益值。分裂增益如式(2.13) 所示，中括号第一项为进行分割后左子节点产生的增益，第二项为分割后右子节点产生的增益，第三项为分割前增益。

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \gamma \tag{2.13}$$

2.5.2　XGBoost的优势

XGBoost算法具有许多避免过拟合的策略，支持交叉验证，支持提前停止构建树从而避免无效计算。XGBoost支持并行训练，训练速度快，异常检测准确，并且可以调整不同样本权重提高模型准确率。

由于XGBoost是有监督异常检测算法并且支持并行计算，训练速度快，模型准确率高。符合本系统迭代重置模型以提高准确率的需求，所以本系统采用XGBoost作为异常检测算法，针对不同数据集训练不同的XGBoost模型。

2.6　LSTM算法

LSTM是一种特殊的RNN神经网络算法，能解决长序列训练过程中的梯度消失和梯度爆炸问题 [43]。相对于RNN，LSTM由于加入了隐藏神经元结构可以长时间记忆时间序列，适合于长序列的趋势预测。

2.6.1　LSTM原理介绍

因为LSTM在每个神经元内部加入输入门、遗忘门和输出门，用于解决长序列训练过程中的梯度消失和梯度爆炸问题。其中每个遗忘门会读取上一时刻输出值h_{t-1}和当前时刻输入值x_t，将两值拼接并通过遗忘门控制保留信息。LSTM内部运算主要有Sigmoid、tanh、加法和乘法等操作，其中Sigmoid和tanh是两种不同的激活函数。Sigmoid函数可以将任意值映射到[0,1]区间，用于决定保留多少信息进入下一记忆单元，其公式为$S(x) = \frac{1}{1+e^{-x}}$。tanh函数是双曲正切函数，可以将任意值映射到[-1,1]区间，其公式为$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$。

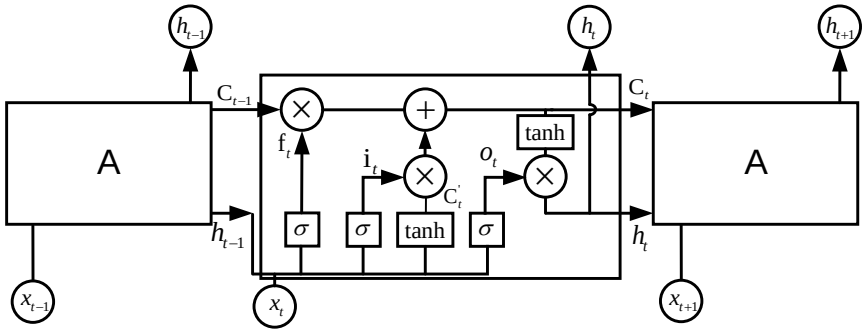

图 2.2: LSTM算法结构图

LSTM算法结构如图2.2所示，LSTM算法中状态保存十分重要，图中C_{t-1}和C_t是状态，它用于长时间记忆需要保留的信息，LSTM算法的具体训练流程如下。

第一步，正向传递。LSTM使用遗忘门决定信息的保留程度f_t，式(2.14)中W_f是遗忘门权重矩阵，b_f是遗忘门的偏移值。这一步将上一轮输出值和当前输入值整合，并且通过遗忘门传入下一记忆单元中：如果遗忘门输出0，则会清除之前的记忆；如果输出为1，则会将之前全部内容记入记忆单元。

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.14}$$

第二步，选择记忆。这一步用于决定单元状态中保存的信息，主要分为生成临时新状态和更新旧状态，图2.2中C'_t是临时新状态，i_t为输入门输出，用于决定上一步的哪些数据需要更新。旧的状态C_{t-1}乘以f_t，决定遗忘哪些信息，保留哪些信息，最后加上新的临时状态C'_t乘以i_t得到新的状态C_t。其中：

$$C'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.15}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.16}$$

第三步，输出。如式(2.17)所示，LSTM首先通过Sigmoid决定输出信息，再通过tanh函数将单元状态映射到[-1,1]，最后乘以Sigmoid门限得到输出值。

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.17}$$

以上步骤中，W_c，W_i，W_o都是权重矩阵。矩阵在最开始迭代时随机生成，在每次向后传递的过程中通过反向传播不断迭代更新，得到准确的矩阵后对前一单元的信息进行保留和更新。

2.6.2　LSTM的优势

LSTM增加了遗忘门，改进了RNN算法，算法结果准确，可以消除长时间序列数据训练过程中梯度消失和梯度爆炸问题，适用于长时间序列趋势预测。

由于LSTM最适合长时间序列问题，而实时监控系统的数据项属于长时间序列，LSTM充分利用长时间序列之间的关系准确预测趋势。所以本系统采用LSTM作为系统趋势预测算法，根据数据集训练LSTM模型后持久化，当需要预测某数据未来趋势时，系统调用相应模型进行趋势预测。

2.7　本章小结

本章主要介绍了系统开发中使用的框架、工具和算法，并分别说明了使用这些框架、工具和算法的原因。本章首先介绍了Zabbix监控产品的组件构成和架构方式，然后介绍了系统使用的Django框架和Redis缓存数据库的功能和优点。最后详细介绍了本系统使用的孤立森林、XGBoost和LSTM三种算法的原理和优点，选择合适的框架、工具和算法可以让系统的功能和性能效果更好。

第三章　智能监控系统的需求分析与设计

本章详细描述基于XGBoost和LSTM的智能监控系统需求分析与设计。首先根据工业场景分析系统功能性需求和非功能性需求，根据需求设计系统整体架构和模块功能。然后详细介绍系统架构和4+1视图，每个功能模块的架构设计、类设计和流程设计。最后介绍系统数据库设计。

3.1　系统整体概述

大量的监控服务器、海量的监控数据和数据间错综复杂的关系极大增加了运维人员的工作压力，运维人员不仅需要为海量数据设置报警阈值，也需要时刻观察资源使用情况，及时对资源进行调度保证系统正常运行。这个过程中设置海量监控指标的阈值是一个机械重复且工作量巨大的任务，数据之间的关联性和系统的差异性也会导致单一阈值不能准确报警，容易出现误报和漏报。

系统旨在提供准确的异常检测和趋势预测功能，从而减少运维人员工作量，提高工作效率并保证系统稳定运行。系统分为四个模块：监控模块的作用是提供详尽的数据采集和报警功能；数据处理模块的作用是处理和管理数据，为异常检测和趋势预测提供服务；异常检测模块的作用是提供准确的异常检测功能；趋势预测模块的作用是提供准确的趋势预测功能。系统采用Django框架开发，利用进程池和Redis缓存等提升系统性能，使用Bootstrap组件和Echarts图等保证前端界面美观性和风格一致性，保证用户使用体验。

3.2　系统需求分析

3.2.1　功能性需求

本系统功能性需求分析主要从监控模块、数据处理模块、异常检测模块和趋势预测模块四部分进行分析。所有功能性需求如表3.1所示。

监控模块主要需求包括监控数据采集、存储和异常报警。采集监控数据时，系统能采集被监控服务器和系统的详细监控指标，将采集的数据按数据来源存储至数据库。异常报警时，系统可以通过微信和邮件实现报警功能，当实时数据被异常检测模块判断为异常后，能及时通过微信企业号和邮件通知用户。

数据处理模块主要需求包括查看异常列表、上传数据集、批量数据标注和查看模型详细信息。查看异常列表时，用户可以按时间和数据来源等条件查看

异常报警信息列表。上传数据集时，系统使用孤立森林对用户上传的数据集进行数据标注预处理。批量数据标注时，用户可以通过多选、单选和全选的方式批量更新数据标签。查看模型详细信息时，用户可以查看异常检测和趋势预测模型详细信息。

异常检测模块主要需求包括训练异常检测模型、重置异常检测模型和实时数据异常检测。训练异常检测模型时，用户选中数据集并训练模型，系统接收请求并使用进程池后台异步训练模型，然后将模型持久化至磁盘和Redis，并通知用户训练完成。用户可以批量更新数据标注并重新训练模型。实时数据异常检测时，异常检测模块接收实时数据，然后调用相应的模型进行异常检测。

趋势预测模块主要需求包括训练趋势预测模型、重置趋势预测模型和查看数据趋势。训练和重置趋势预测模型与异常检测模型相似，这里不再累述。查看数据趋势时，用户选择数据集进行趋势预测，系统接收用户请求并调用相应的趋势预测模型进行预测，并将预测结果通过折线图展示。

表 3.1: 功能需求列表

需求编号	需求名称	需求描述
R1	采集监控数据	系统能自动采集被监控服务器各种监控指标数据
R2	异常报警	系统在检测到实时数据异常后通过微信和邮件通知用户
R3	查看异常列表	用户可以按时间、数据来源等多条件查看历史异常报警信息列表，可以查看详细的报警时间和监控项信息
R4	上传数据集	用户可以选择异常比例并上传训练模型需要的数据集，系统对数据集进行数据标注预处理并存储至数据库中
R5	批量数据标注	用户可以查看数据信息，在发现系统表现与实时数据展示不一致时可以批量进行数据标注
R6	查看模型详细信息	1.用户可以查看异常检测模型的精确率、精确率、已经训练的数据量、模型创建时间、最后更新时间、是否训练完成等信息 2.用户可以查看趋势预测模型的准确率、模型创建时间、最后更新时间等信息
R7	训练模型	用户可以训练多个不同的异常检测和趋势预测模型，每个模型对应不同的监控指标
R8	重置模型	系统监控指标发生变化或者模型准确率降低时，用户可以重置对应的异常检测模型和趋势预测模型来适应新的监控指标
R9	实时异常检测	对监控服务器产生的实时监控指标进行异常检测
R10	查看数据趋势	用户可以查看某数据指标未来一段时间趋势

3.2.2　非功能性需求

基于XGBoost和LSTM的智能监控系统主要目标是提供准确的异常检测与趋势预测功能。系统可以及时对被监控系统的实时数据检测异常并报警，准确预

测数据未来趋势走向。在满足系统功能性需求的前提下，保证系统及时性和可靠性十分重要。因此本小节从系统使用者的视角分析本系统非功能性需求。

（1）及时性。本系统应该及时对采集的数据进行异常检测，如果有异常则应该及时通知相关运维人员，以便运维人员及时掌握系统故障并处理。

（2）扩展性。为方便后期系统加入更多的功能和业务，系统功能模块间耦合度应该尽可能小。对于基础功能，系统应该预留接口，方便未来业务扩展。同时，随着监控指标的增长、系统所需硬件资源和网络资源应该能够扩展以满足系统异常检测和趋势预测需求。

（3）可靠性。系统在模型训练完成后应该将模型对象备份，如出现系统异常宕机或者重启的情况时，系统能够将原有模型对象恢复以保证实时数据异常检测和数据趋势预测的正确性，保证系统可靠稳定运行。

3.2.3　系统用例描述

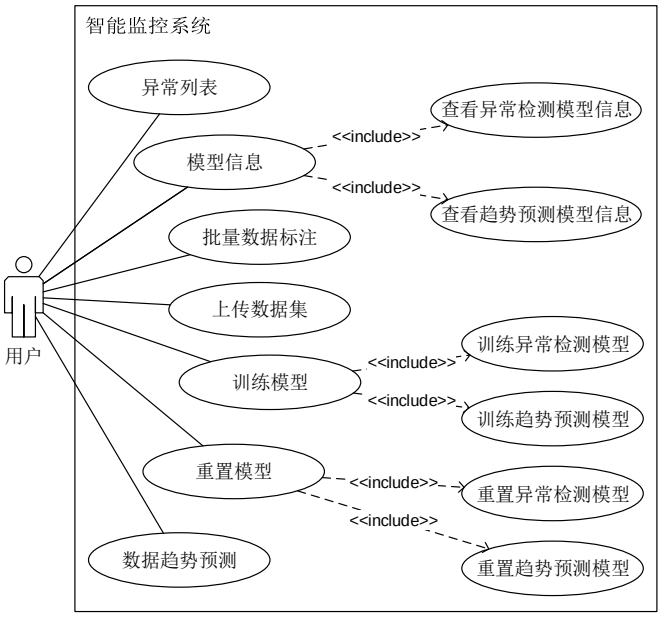

图 3.1: 系统用例图

根据上述功能性需求分析，系统用例图如图3.1所示。本系统主要用户是运维人员，用户可以查看异常列表、上传数据集、进行批量数据标注、对数据进行趋势预测、查看模型信息、训练模型和重置模型。其中查看模型信息包括查看异常检测和趋势预测两类模型详细信息，训练模型与重置模型包括训练与重置异常检测和趋势预测两类模型。本小节将详细介绍系统用例图中7个用例的前置条件、后置条件、触发条件、优先级和正常流程等信息。

表 3.2: 异常列表用例描述

ID	**C1**
名称	异常列表
参与者	参与者：用户 目标：用户查看异常信息列表
触发条件	用户在“异常信息”页面查看异常列表
前置条件	系统已收集异常信息并正常运行
后置条件	无
优先级	中
正常流程	1.用户点击“异常信息”菜单栏 2.系统展示异常信息列表，每条数据有详细的时间和数据来源等信息

异常列表用例描述如表3.2所示。异常列表用于集中展示被监控系统的异常信息。被监控系统或应用产生实时数据后，异常检测模块接收实时数据并进行异常检测，如果判断为异常则将数据集中存储。用户可以通过搜索和排序等方式查看异常信息，以便快速了解系统异常。

表 3.3: 上传数据集用例描述

ID	**C2**
名称	上传数据集
参与者	参与者：用户，系统 目标：用户上传数据集，系统使用孤立森林对数据进行数据标注预处理并存储
触发条件	用户在“上传数据集”页面选择文件并点击“上传”按钮
前置条件	系统正常运行
后置条件	无
优先级	高
正常流程	1.用户在“上传数据集”页面点击“选择文件”按钮 2.选择待上传文件，选择该数据集异常比例（默认0.1），并点击“上传”按钮 3.系统接收文件，获取数据信息和异常比例传入孤立森林 4.孤立森林算法依据异常比例对数据集进行数据标注预处理 5.将预处理过带有标签的数据存储至数据库中 6.提示用户上传成功
扩展流程	3a.系统发现该数据集已上传 1.系统提示用户该数据集已上传，是否继续覆盖上传、合并上传或者取消上传 1a.用户选择覆盖上传或者合并上传 1.返回正常流程第3 步 1b.用户选择取消上传 1.系统取消上传数据集

上传数据集用例描述如表3.3所示，用于描述用户上传数据集并进行数据标注预处理的过程。用户筛选监控模块采集的数据，根据历史情况设置异常比例

并上传系统，系统使用异常比例和孤立森林算法对上传的数据集进行数据标注预处理并存储，数据标注预处理可以极大减少用户数据标注工作量。数据是异常检测和趋势预测的基础，所以上传数据集优先级为高。

表 3.4: 批量数据标注用例描述

ID	**C3**
名称	批量数据标注
参与者	参与者：用户 目标：用户通过批量标注更正数据标签，提高数据标注的准确率，从而提高异常检测模型的准确率
触发条件	用户在“数据标注”页面批量选择数据进行数据标注
前置条件	系统正常运行，数据集已经上传
后置条件	无
优先级	高
正常流程	1.用户进入“数据标注”页面 2.用户选择数据集、时间区间和标签后点击“查询”按钮 3.系统按用户选择的条件查询数据库并返回数据，前端页面使用表格展示数据 4.用户通过多选、单选和全选的方式批量选择数据 5.用户点击“标签置为0”或者“标签置为1”按钮 6.系统根据用户选择的标签重置用户选中的数据 7.系统提示更新成功。
扩展流程	4a.用户选择标签后直接点击“全部更改”按钮 1.系统根据用户选择的数据集、时间区间和标签重置该数据集这段时间区间内所有数据标签

批量数据标注用例描述如表3.4所示。批量数据标注提供界面让用户可以批量对数据库中数据标签进行更改，数据集经过孤立森林预处理数据标注后存入数据库，异常检测模型训练完成后，对实时数据检测后也会将实时数据和标签存入数据库。用户可以按数据集名称、时间区间和标签多条件查看数据，如果发现数据标注预处理或者异常检测错误的地方，可以通过多选、单选和全选的方式批量更改数据标签。系统异常一般是一段时间区间，所以系统提供将数据集某段时间区间内所有数据批量更改标签的功能，如表3.4扩展流程所示。

模型信息用例描述如表3.5所示。模型信息用于提供异常检测和趋势预测所有模型信息查看功能，用户可以点击“异常检测模型信息”和“趋势预测模型信息”子菜单栏分别查看两类模型信息。“异常检测模型信息”界面展示所有异常检测模型的数据来源、模型名称、精确率、召回率、f1值、已训练数据量、是否需要重新训练、模型创建时间和最后修改时间信息。页面表格根据f1值显示不同颜色，用于提醒用户异常检测模型是否准确。“趋势预测模型信息”界面展示所有趋势预测模型的数据来源、模型名称、模型均方根误差、模型创建时

间和模型最后修改时间。页面表格根据模型均方根误差显示不同颜色，用于提醒用户趋势预测模型是否准确。

表 3.5: 模型信息用例描述

ID	**C4**
名称	模型信息
参与者	参与者：用户 目标：用户查看异常检测和趋势预测模型的详细信息
触发条件	用户进入“异常检测模型信息”或“趋势预测模型信息”页面
前置条件	系统正常运行
后置条件	无
优先级	高
正常流程	1.用户进入“模型信息”页面 2a.用户点击“异常检测模型信息”子菜单 3a.系统展示所有异常检测模型的详细信息，包括数据来源、模型名称、精确率、召回率、f1 值、训练数据量、模型创建时间和模型最后修改时间信息 2b.用户点击“趋势预测模型信息”子菜单 3b.系统展示所有趋势预测模型的详细信息，包括数据来源、模型名称、模型均方根误差、模型创建时间和模型最后修改时间
特殊需求	3a.根据每个异常检测模型f1值的大小将单元格填充成红、橙、绿色，分别表示模型准确率低、中、高。用来提示用户是否应该重新训练模型提高准确率 3b.根据每个趋势预测模型均方根误差将单元格填充为红、橙、绿色，分别表示模型准确率低、中、高。用来提示用户是否应该重新训练模型提高准确率

表 3.6: 训练模型用例描述

ID	**C5**
名称	训练模型
参与者	参与者：用户 目标：用户根据上传的数据集训练异常检测或趋势预测模型
触发条件	用户在“训练模型”页面选择模型类型和数据集，然后点击“训练模型”按钮
前置条件	系统正常运行，数据集已经上传
后置条件	无
优先级	高
正常流程	1.用户进入“训练模型”页面，选择模型类型和数据集，点击“训练模型”按钮 2.系统根据模型类型和数据集名称后台异步训练模型，训练完成后将模型持久化到磁盘中和Redis中，将模型信息存储至数据库中 3.系统提示模型训练成功
异常流程	2a.数据集的数据量较少，不能得出准确的模型，系统提示数据不足

训练模型用例描述如表3.6所示，用于提供模型训练并持久化功能。用户选择模型类型和数据集并开始训练，系统接收模型类型和数据集，从数据库获取

数据集信息，并调用相应算法后台异步训练模型。训练完成后系统将模型持久化到磁盘并缓存至Redis，将模型信息存储至数据库并提示用户训练成功。

表 3.7: 重置模型用例描述

ID	**C6**
名称	重置模型
参与者	参与者：用户 目标：由于被监控数据项变更等原因导致模型准确率降低时，用户可以重置模型以提高模型准确率
触发条件	用户在模型信息界面选择模型并点击“重置模型”按钮
前置条件	系统正常运行
后置条件	无
优先级	高
正常流程	1.用户进入“异常检测模型信息”或者“趋势预测模型信息”页面 2.查看模型详细信息，选中要重置的模型并点击“重置模型”按钮 3.系统根据该模型对应的数据库表中数据和标签，后台异步重新训练该模型 4.训练完成后将模型对象持久化到磁盘和Redis，并更新数据库中模型信息 5.系统提示模型重置成功

重置模型用例描述如表3.7所示。重置模型包括重置异常检测模型和趋势预测模型。当孤立森林数据标注不准确或者监控数据项发生变化导致异常检测模型准确率降低时，用户可以通过批量数据标注，然后在“异常检测模型信息”或者“趋势预测模型信息”页面点击“重置模型”按钮重置相应模型。通过迭代重置模型的方式不断提高模型准确率，所以重置模型的优先级高。

表 3.8: 数据趋势预测用例描述

ID	**C7**
名称	数据趋势预测
参与者	参与者：用户 目标：用户使用趋势预测模型预测某数据未来30个时间点的数据值
触发条件	用户在“趋势预测”页面选择要预测的监控数据项并点击“预测”按钮
前置条件	系统正常运行，趋势预测模型训练完成
后置条件	无
优先级	高
正常流程	1.用户进入“趋势预测”页面 2.用户选择要预测的数据集，点击“预测”按钮 3.系统根据用户选择的数据集，获取该数据集最后50个数据传给对应的已经训练完成的趋势预测模型 4.趋势预测模型根据历史数据预测未来30个时间点的数据值 5.系统将预测值存入数据库并通过折线图展示给用户

数据趋势预测用例描述如表3.8所示，主要用于数据未来趋势预测。用户在“趋势预测”界面选中数据集并点击“预测”按钮，系统根据数据集获取历史数据和相应趋势预测模型预测未来趋势，并通过折线图展示趋势。

3.3　系统总体设计

本系统是一个结合Zabbix、孤立森林、XGBoost和LSTM的智能异常检测与趋势预测监控系统。系统总体架构如图3.2所示，系统分为监控模块、数据处理模块、异常检测模块和趋势预测模块，用户通过前端与系统交互，系统使用Redis和磁盘持久化模型对象。

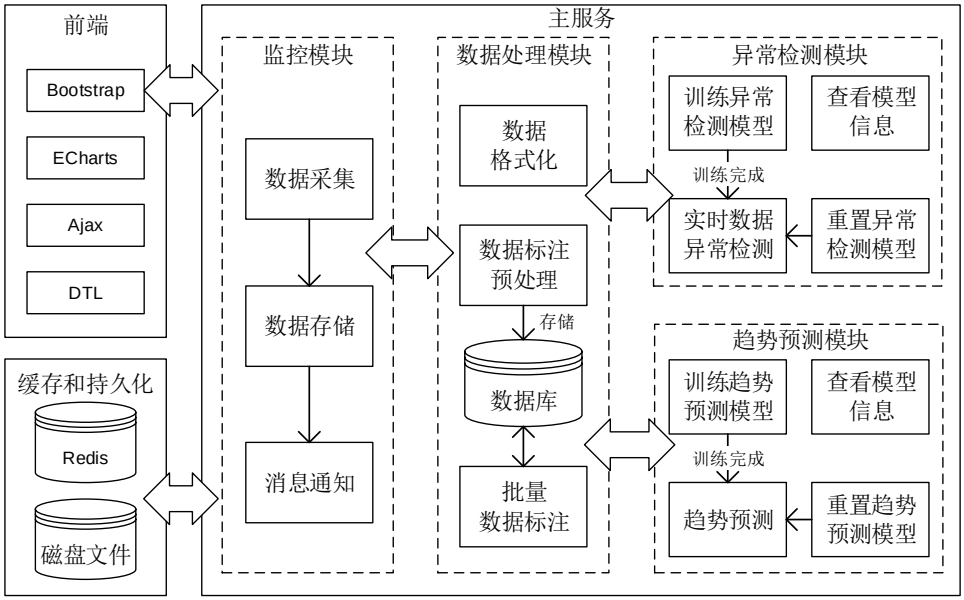

图 3.2: 系统总体架构图

监控模块主要功能是数据采集存储和消息通知。模块使用Zabbix采集监控数据，过滤需要的数据集并存储，使用Python脚本完成微信和邮件报警功能。

数据处理模块主要功能包括指定异常比例上传数据集、使用孤立森林完成数据标注预处理、格式化处理数据、查看数据集和批量进行数据标注等。

异常检测模块主要功能包括查看异常检测模型信息，异常检测模型的训练、使用和重置。用户可以根据数据集训练异常检测模型，可以通过批量数据标注然后重置模型的方式提高模型的准确率，可以查看所有异常检测模型详细信息。异常检测模块接收系统实时数据，并调用相应模型判断数据是否异常，如异常则通过监控模块消息通知功能报警并展示异常数据。

趋势预测模块主要功能包括查看趋势预测模型信息，趋势预测模型训练、预测和重置。用户可以训练、重置模型，可以预测监控项数据趋势走向并使用折线图展示预测数据，可以查看所有趋势预测模型的详细信息。

本系统前端使用Django模板引擎DTL快速渲染，配合使用Bootstrap保证页面简洁美观、风格一致，使用Echarts图表展示趋势预测折线图，前后端页面请求采用HTTP请求方式，页面内使用Ajax异步请求，保证页面相应速度。

在性能方面，系统通过Redis缓存提高性能。大量的模型同时存放在内存中会导致大量内存浪费，所以系统考虑使用Redis和磁盘存储模型对象。模型训练完成后，系统将模型对象缓存至Redis并设置过期时间为2小时，同时持久化到磁盘。当进行异常检测和趋势预测需要使用模型对象时，系统首先判断模型对象是否存在Redis中，如果存在则直接从Redis中获取并使用，如果不在则从磁盘中读取，然后缓存至Redis中并设置过期时间。缓存数据集与缓存模型对象类似，上传数据集后通过Redis缓存数据集名称，再次上传前先通过Redis判断该数据集名称是否已存在，如果已存在则提示用户选择覆盖、增量或者取消上传。

下面从场景视图、逻辑视图、开发视图、进程视图和物理视图五个方面详细介绍本系统总体结构。场景视图是从用户的角度出发，识别业务需求，描述业务场景，是设计的起点，与UML中用例图对应，本系统用例图如图3.1所示。

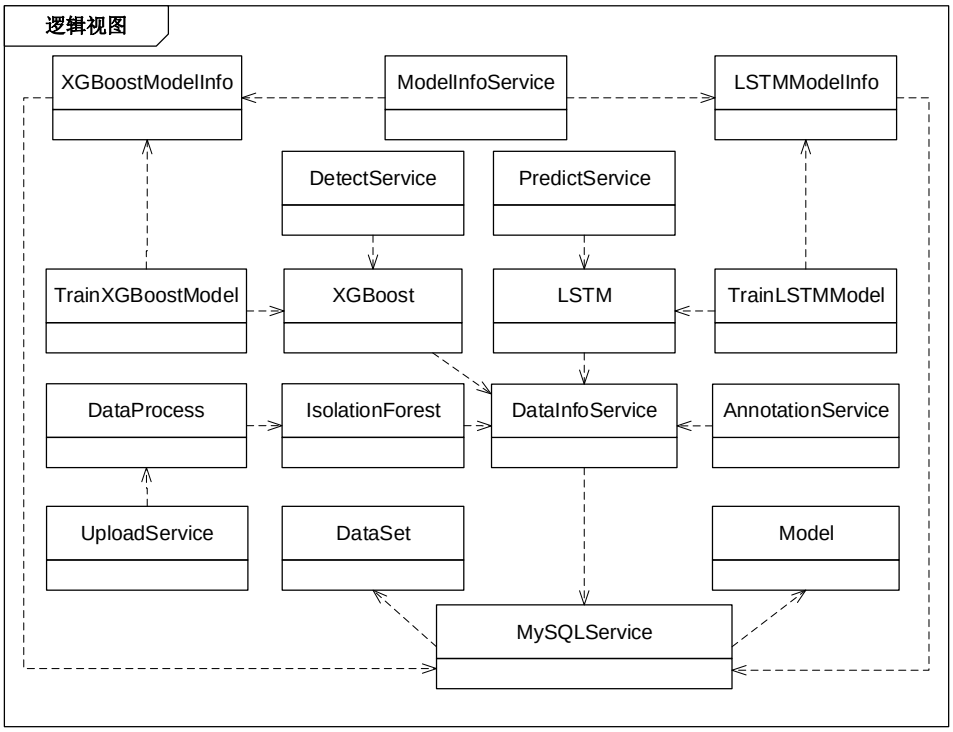

图 3.3: 逻辑视图

逻辑视图从用户的角度描述系统为用户提供的功能和服务，可以使用UML中的类图来描述。图3.3是系统UML类图，图中UploadService提供上传数据集的功能，并调用IsolationForest完成数据标注预处理。AnnotationService为用户提供批量数据标注功能，DataInfoService为用户提供数据操作功能，包括多条件查询数据集和标签、批量更新数据标签等功能，DataProcess用于处理模型数据。TrainXGBoostModel为用户提供训练和重置异常检测模型的功能，训练时调用XGBoost类训练模型，训练完成后存储模型信息，DetectService在异常检测模型训练完成后提供实时数据异常检测功能。TrainLSTMModel提供训练和重置趋势预测模型的功能，PredictService在趋势预测模型训练完成后调用对应的LSTM类预测监控项未来趋势。ModelInfoService提供异常检测和趋势预测模型详细信息管理功能，XGBoostModelInfo提供异常检测模型信息管理功能，LSTMModelInfo提供趋势预测模型信息管理功能。MySQLService用于数据库操作。DataSet和Model分别对应数据集和模型信息。

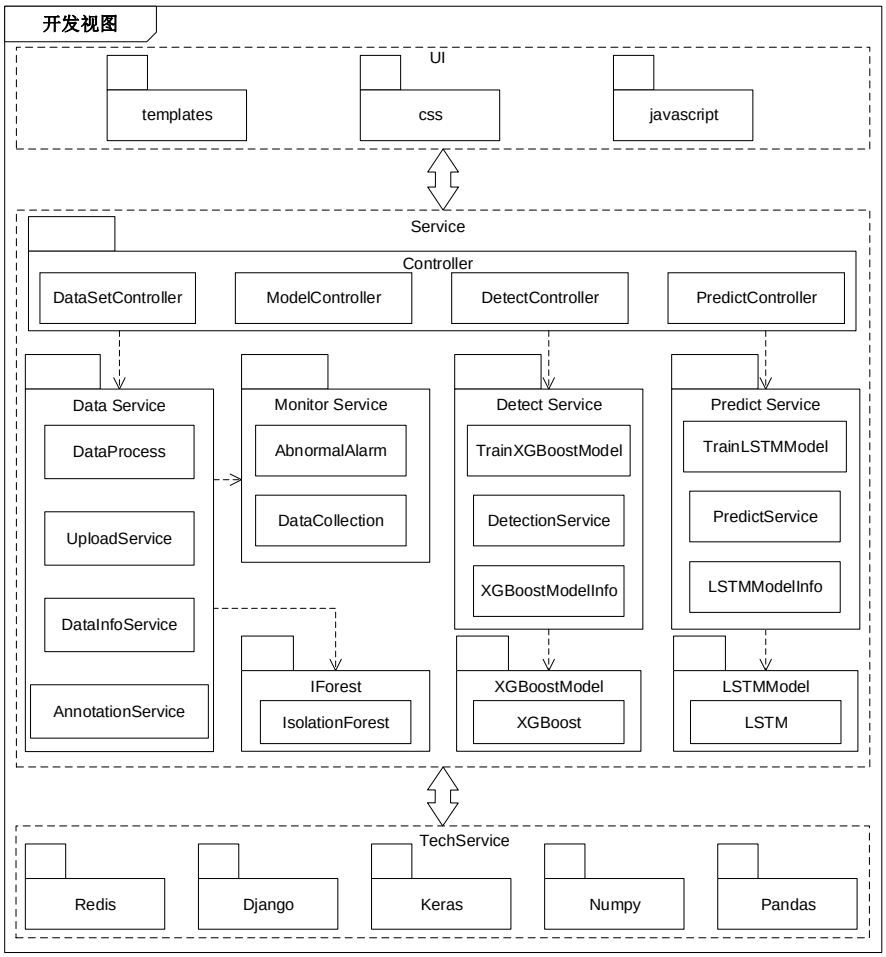

图 3.4: 开发视图

开发视图是从开发人员的角度，描述在开发环境下软件的静态组织结构。本系统开发视图如图3.4所示，UI部分由templates模板、CSS和JavaScript组成。服务端根据系统功能和模块划分，Controller包负责接收前端请求，Monitor Service包负责监控数据采集和异常报警，Data Service包负责监控数据处理和管理，Detect Service包负责实时数据异常检测，Predict Service包负责数据趋势预测，IForest包负责实现孤立森林算法，XGBoostModel包负责实现异常检测算法，LSTMModel包负责实现趋势预测算法。技术服务主要涉及Redis缓存、Django框架、Keras深度学习库、Numpy和Pandas工具库。

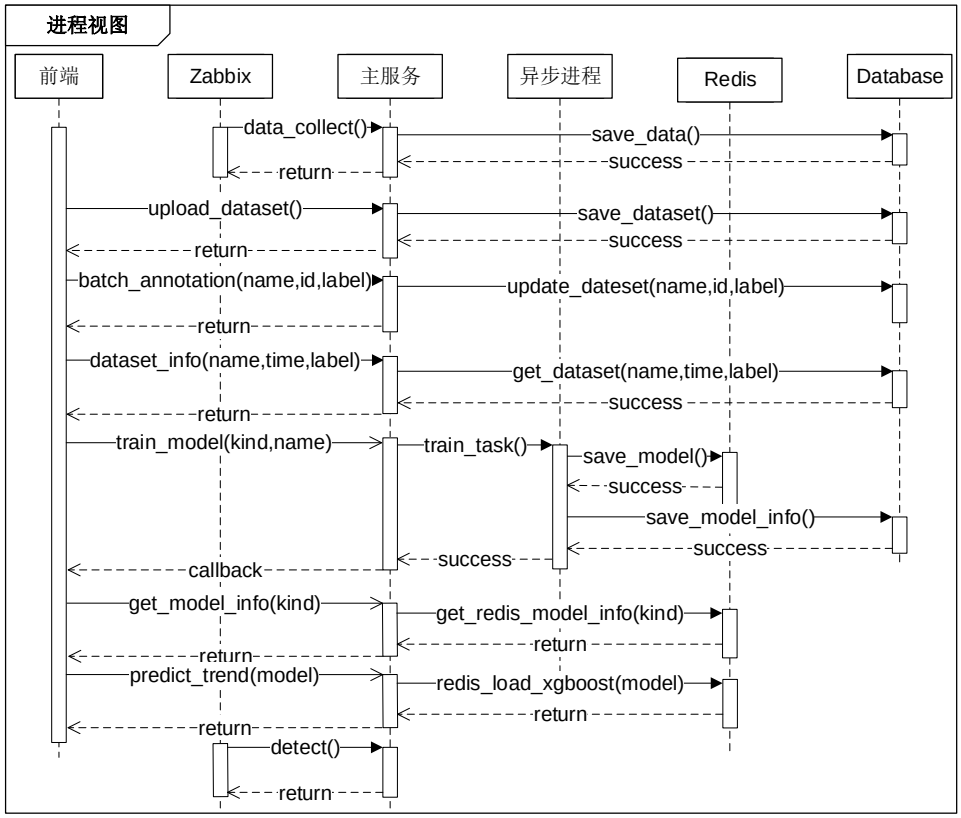

图 3.5: 进程视图

进程视图是描述系统对并发和同步的设计，包括系统进程、线程和通信等。本系统进程视图如图3.5所示，Zabbix进程会主动采集数据并传递给主服务。前端可以与主服务通信完成上传数据集、批量数据标注、训练模型、查看模型信息、查看数据信息和数据趋势预测功能。主服务与Redis通信完成模型对象的缓存，主服务与Database通信完成数据集和模型信息的存储。训练模型是一个异步过程，主服务会调度进程池异步完成训练任务。Zabbix进程采集实时数据，同步调用主服务对实时数据进行异常检测。

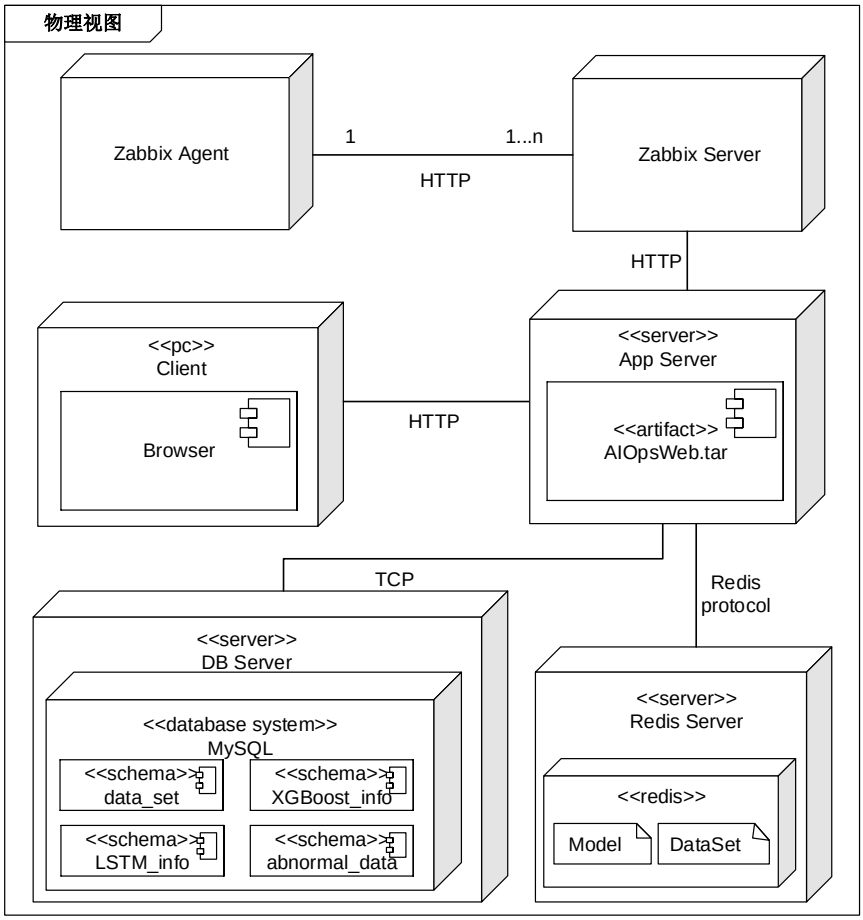

图 3.6: 物理视图

物理视图是从部署的角度描述软硬件的映射关系。本系统物理视图如图3.6所示，多个Zabbix Agent服务器将采集到的数据通过HTTP传递给Zabbix Server服务器，Zabbix Server服务器通过HTTP将实时数据传给应用服务器完成异常检测。用户通过浏览器对应用服务器进行访问，完成数据通信和处理，应用服务器通过TCP传递SQL给数据库服务器，完成数据增删改查操作。应用服务器通过与Redis服务器通信完成数据和模型的缓存操作。

3.4　监控模块设计

3.4.1　架构设计

监控模块架构图如图3.7所示，Zabbix Client通过多种方式采集数据后传给Zabbix Server，Server端接收数据并存储至数据库中。监控模块通过配置脚本决定采集哪些监控数据项以及数据采集频率，通过编写Python脚本实现微信企

业号和邮件异常报警功能。该模块使用Agent、ICMP、自定义脚本和自定义计算公式的方式采集信息。监控模块主要功能包括监控数据采集、数据存储和消息通知。监控模块必须采集准确且详细的数据作为系统异常检测与趋势预测等功能的支撑。当数据被判定为异常时，监控模块能够通过微信企业号和邮件通知相关运维人员。

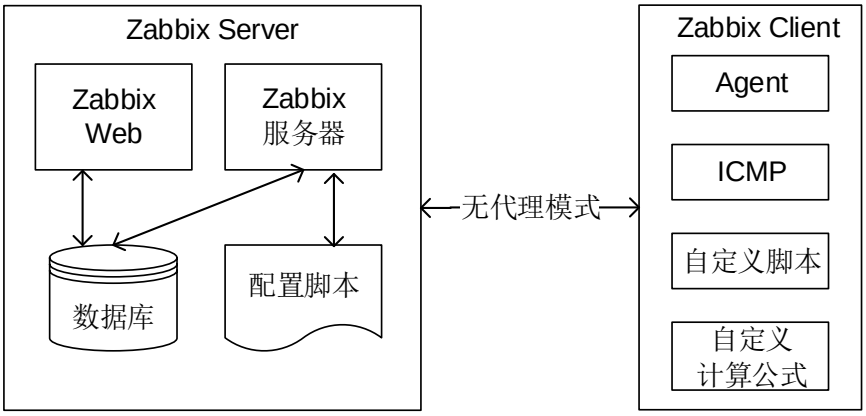

图 3.7: 监控模块架构图

数据采集功能负责采集被监控服务器的各种资源信息和应用信息。这些信息来自不同的地区、机房和服务器终端。本系统采用Agent、ICMP和自定义脚本的方式进行信息采集，其中Agent用于获取服务器基本信息，ICMP用于获取网络情况，自定义脚本用于获取网站应用等监控信息，自定义计算公式用于获取颗粒度更细的数据。数据采集类型包括服务器基本信息、服务信息和应用信息，例如CPU使用率、内存使用量、磁盘使用率、系统文件数、实时进程数、实时线程数、MySQL运行状态、网站状态、网站在线用户数、网站用户总数、每日用户注册量和每日用户登录量等信息。针对不同的服务器、服务和应用，应该编写不同脚本采集数据信息，并根据具体监控项设置不同的采集频率。

数据存储用于存储采集的所有数据，系统采用MySQL作为Zabbix数据库。存储的数据包括服务器数据、应用数据、用户账号信息、监控项阈值、报警方式和监控项触发器等。

消息通知功能用于当系统出现异常时使用微信和邮件通知用户，系统通过编写Python脚本的方式实现消息通知功能。用户可以制定报警条件、报警严重等级、报警时间段和报警信息等。当异常检测模块判断实时数据为异常数据时，系统会根据用户指定的报警方式进行报警。本系统采用微信和邮件报警方式通知用户，其中微信通过微信企业号群发通知关注企业号的人员。

3.4.2　数据采集和报警流程设计

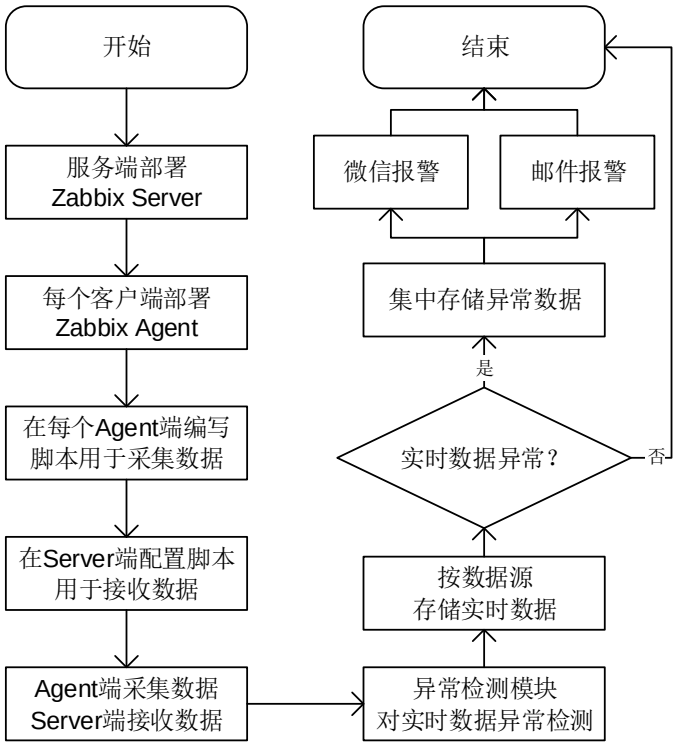

图 3.8: 数据采集和报警流程图

数据采集和报警流程图如图3.8所示，首先选择一台服务器部署Zabbix Server，在每台被监控服务器上部署Zabbix Agent，然后在每个Agent端配置脚本完成相关数据项监控和采集，在Server端通过配置脚本收集每个Agent端采集的数据并存储。当Agent端产生实时数据后，Server端接收数据，然后通过异常检测模块对实时数据进行异常检测，并按数据来源存储，如果发现异常则将异常数据集中存储至数据库中，并通过微信企业号和邮件通知相关运维人员。

3.5　数据处理模块设计

3.5.1　架构设计

数据处理模块架构图如图3.9所示，其功能包括用户上传数据集、使用孤立森林对数据集进行数据标注预处理、按数据来源和时间区间等多条件查询数据集和模型信息、批量数据标注和数据格式处理。数据处理模块的主要功能是对监控模块采集的数据进行格式处理和管理，为异常检测模块和趋势预测模块提供数据服务。

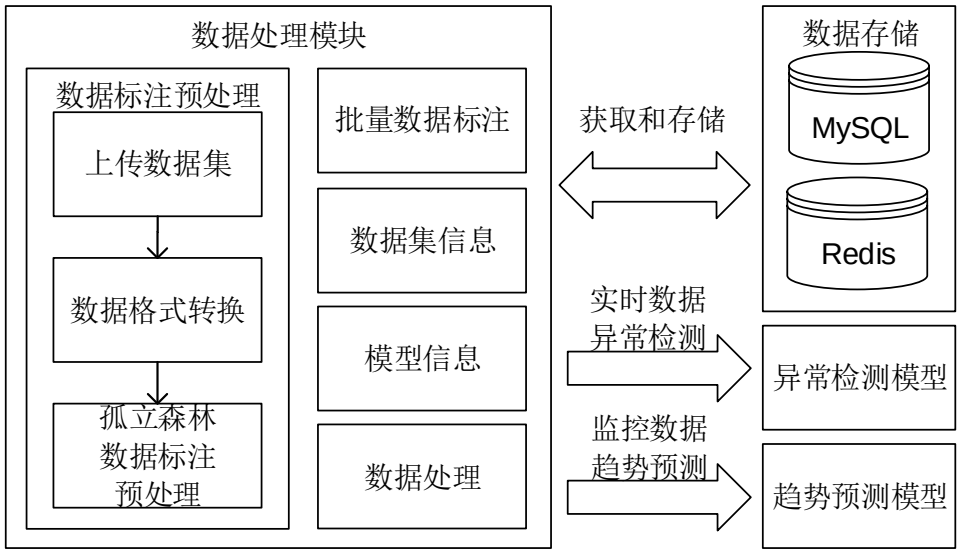

图 3.9: 数据处理模块架构图

数据标注预处理中。系统首先对用户上传的数据集进行格式转换，将csv文件内容转换成Numpy ndarray对象（Numpy ndarray是一种多维数组结构，用于存储同一类型对象），以便于被孤立森林模型识别。然后使用孤立森林对每条数据进行数据标注预处理，并将数据和标签一一对应存储至数据库中，将数据集名称缓存至Redis中。当用户重复上传数据集时，通过Redis中数据集名称可以快速判断是否重复上传。

批量数据标注主要是按照用户在前端页面选定的条件更改数据标签，用户可按数据来源、时间区间和标签等多条件查询数据，通过单选、多选和全选等方式选中数据并更新数据标签。

用户查看模型与数据集信息。当用户查看模型信息时，系统通过判断每个模型的准确率提示用户是否需要重新训练模型，例如异常检测模型以“f1”值（精确率和召回率调和平均）作为判断模型准确率的依据，趋势预测模型以“均方根误差”（预测值和真实值误差）作为判断模型准确率的依据。用户还可以查看每条数据的获取时间、数据来源和数值等信息。

数据处理主要包括两个方面：一方面是对监控模块采集的实时数据处理，首先转换实时数据格式，通过判断实时数据的来源找到对应的异常检测模型，并将数据传给该异常检测模型进行异常检测；另一方面。在用户选择数据集并进行趋势预测后，系统从数据库中获取对应数据集的最新50条数据，过滤掉数据标签后传给对应的趋势预测模型进行趋势预测。

3.5.2　类图设计

数据处理模块类图如图3.10所示，UploadService通过解析用户上传的文件完成上传数据集功能。DataProcess用于处理孤立森林、XGBoost和LSTM算法模型所需数据的格式，让数据能被模型识别。IsolationForest使用孤立森林对已格式化的数据进行数据标注预处理。ModelInfoService用于管理异常检测和趋势预测模型信息。AnnotationService提供批量数据标注功能。DataInfoService用于查询数据集信息。MySQLService用于提供数据处理模块与数据库交互功能。

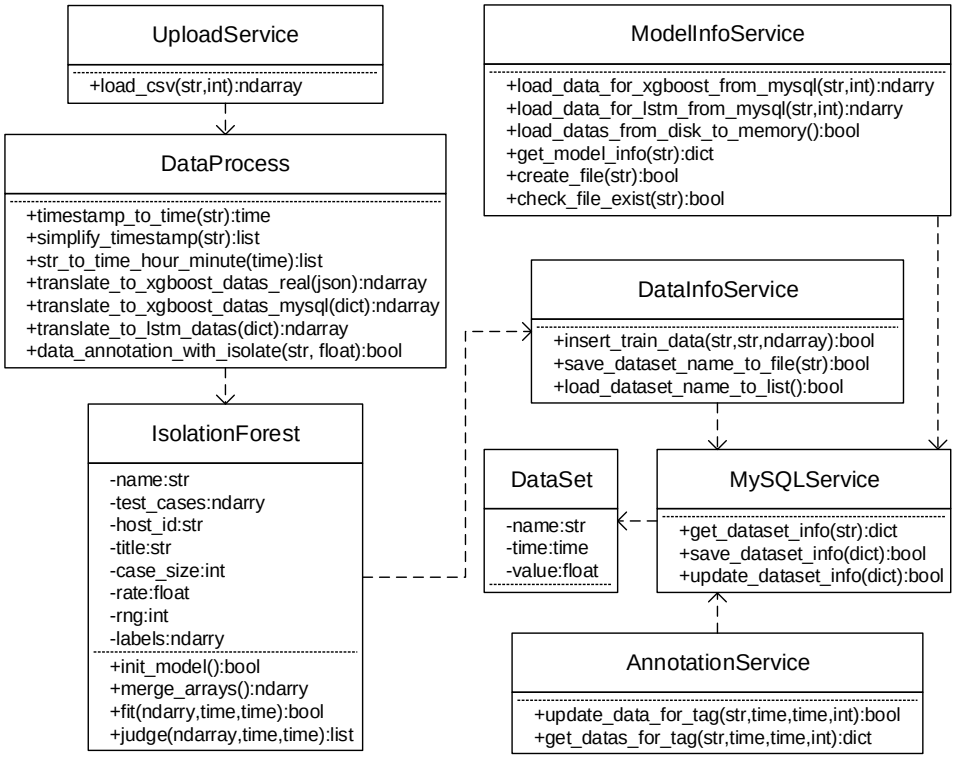

图 3.10: 数据处理模块类图

3.5.3　上传和数据处理流程设计

上传和数据处理流程如图3.11所示，在用户上传数据集时，系统会先检索Redis判断数据集是否已上传：如果没有上传，则先解析文件并格式化数据，然后将数据传递给孤立森林进行数据标注，最后将数据和标签存储至数据库并将数据集名称缓存至Redis；如果数据集已存在则提示用户该数据集已上传，并提示用户选择覆盖上传、增量上传或者取消上传。如果用户选择覆盖上传，则清空该数据集历史数据并重新写入数据，如果选择增量上传则直接添加数据至数据库，如果用户选择取消上传，系统终止用户上传操作。

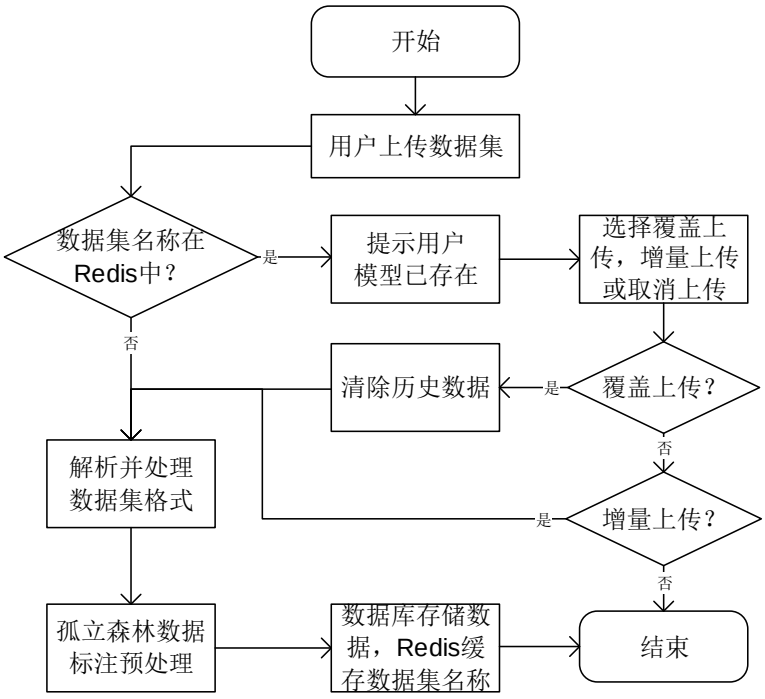

图 3.11: 上传和数据处理流程图

3.6　异常检测模块设计

3.6.1　架构设计

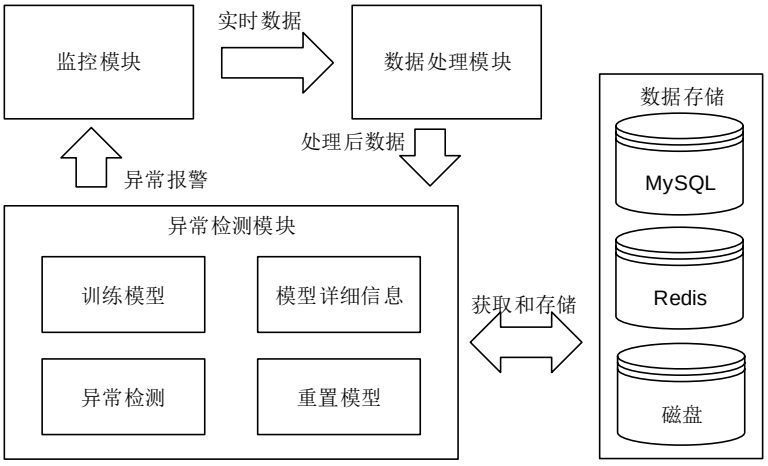

图 3.12: 异常检测模块架构图

异常检测模块架构图如图3.12所示，异常检测模块的任务是对被监控系统的实时数据进行异常检测，并且允许用户通过前端界面查看异常检测模型详

细信息，训练和重置异常检测模型。异常检测模块对实时数据进行异常检测时，如发现异常则通过监控模块消息通知功能报警。异常检测模块采用有监督的XGBoost作为异常检测算法，用户可以针对不同监控项训练多个相互独立，互不影响的异常检测模型。

训练模型时，用户选择数据集并训练，系统根据数据集从数据库中获取相应的数据信息，此时数据库表中每条数据都已经过数据标注预处理，都具有标签，可以被XGBoost算法使用。系统根据数据和标签使用进程池异步后台训练异常检测模型，提高系统效率和性能。

模型训练完成后系统将模型详细信息存储至数据库，将模型名称和模型对象缓存至Redis，将模型对象持久化到磁盘。用户可以通过前端界面获取模型详细信息，当用户重复训练模型时，系统会先判断该模型名称是否在Redis中，如果存在则提示模型已存在，如果不存在则开始训练模型。

由于用户可能会训练大量的模型，将所有模型存放在内存会导致大量内存浪费，所以系统采用Redis和磁盘共同存储模型对象的方式，合理使用内存资源，提升模型对象读取速度。Redis在缓存模型对象时需要设置过期时间为2小时，避免模型对象长期存于内存造成浪费。

异常检测时，监控模块将采集的实时数据传给异常检测模块，系统根据实时数据的来源找到相应的异常检测模型，然后使用该模型对实时数据进行异常判断，并存储实时数据和标签，如果是异常数据则通过监控模块报警。

当用户选择重置某模型时，系统直接获取该模型对应数据集最新数据进行训练，训练完成后更新Redis、磁盘和数据库信息。

3.6.2　类图设计

异常检测模块类图设计如图3.13所示，TrainXGBoostModel类主要负责训练和重置异常检测模型。RedisService负责对Redis数据库操作，包括模型训练完成后存储模型、重置模型时更新Redis、删除模型和查看模型是否已经存在。XGBoostModelInfo负责异常检测模型详细信息管理功能，用户通过XGBoostModelInfo查看所有异常检测模型详细信息。PersistenceService负责对异常检测模型持久化到磁盘和加载模型到内存。DetectService负责接收实时数据并从Redis和磁盘中获取相应模型进行异常检测。XGBoost是算法模型类，存储每个模型必备的参数信息和方法，每次训练和重置模型时都会初始化一个XGBoost实例。MySQLService负责异常检测模块与数据库交互功能。上述各类共同组成异常检测模块，协作完成训练异常检测模型、重置异常检测模型、查看异常检测模型信息和实时数据异常检测功能。

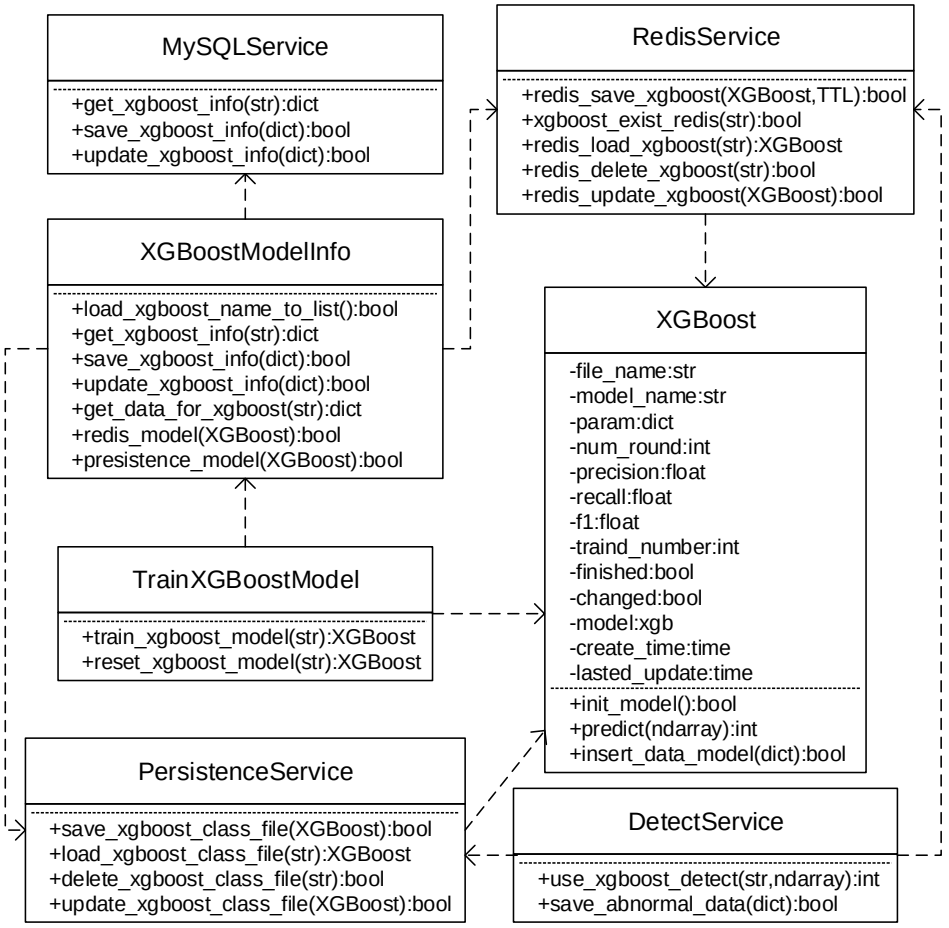

图 3.13: 异常检测模块类图

3.6.3　异常检测流程设计

异常检测流程图如图3.14所示，异常检测流程图主要分为训练模型、重置模型和实时数据异常检测三个具体流程。模块接收请求后，先判断是否是训练和重置模型请求，如果是重置模型请求则根据用户选中的数据集直接从数据库中获取最新数据，经过格式处理后传给XGBoost算法模型进行异步训练，训练时调用进程池异步后台训练，训练完成后将模型对象持久化到磁盘和Redis，并将模型名称持久化至Redis。如果是训练模型请求则先判断Redis中是否存在该模型名称，如果模型名称已存在则提示用户重复训练，如果不存在则开始后台异步训练模型。如果是实时数据异常检测请求，则根据实时数据的来源调用相应异常检测模型进行异常检测，并将检测的结果存储至数据库。

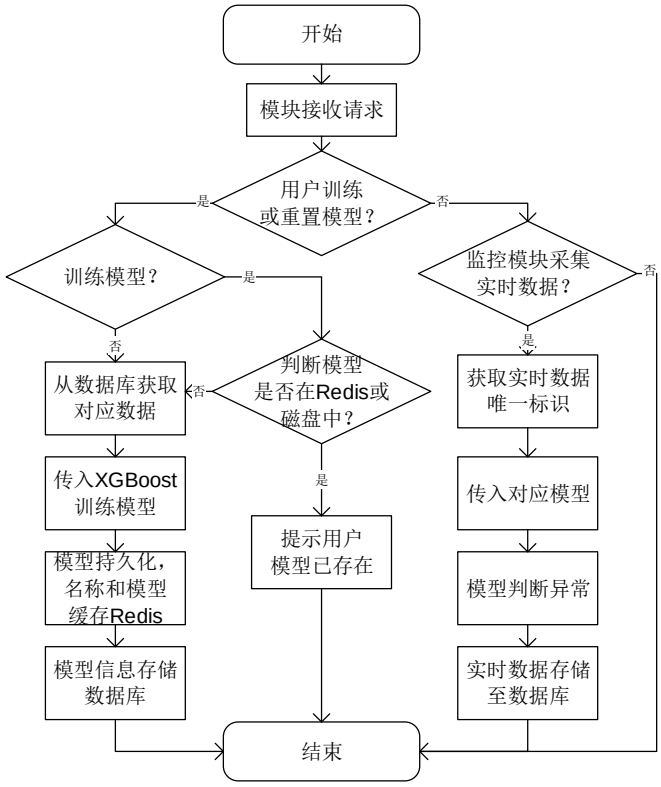

图 3.14: 异常检测流程图

3.7　趋势预测模块设计

3.7.1　架构设计

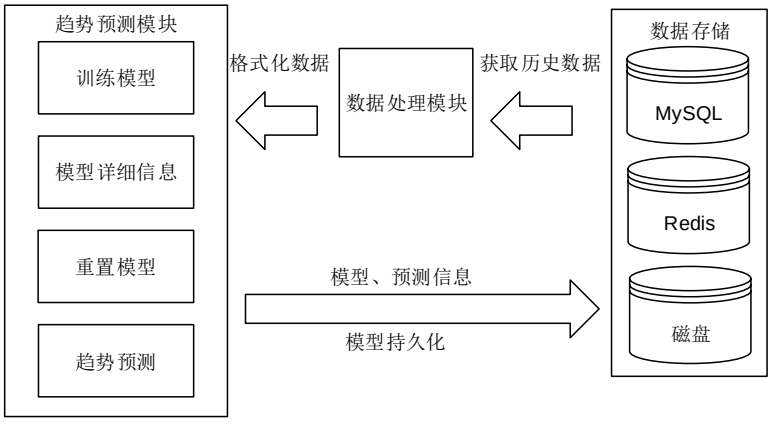

图 3.15: 趋势预测模块架构图

趋势预测模块架构图如图3.15所示，趋势预测模块的任务是预测某监控项指标未来30个时间点的趋势走向。趋势预测模块允许用户在前端训练模型、触

发预测、查看预测趋势、查看预测模型详细信息和重置模型等，模块主要使用LSTM神经网络模型完成趋势预测功能。

当用户选择数据集并训练模型时，系统首先根据数据集从数据库中获取相应的数据，去掉所有数据标签并格式化，以便被LSTM算法模型识别和使用。然后系统从进程池中获取一个空闲进程进行后台异步模型训练。训练完成后将模型信息存储至数据库，将模型对象持久化到磁盘，将模型名称和模型对象缓存至Redis并设置过期时间。

当用户进行趋势预测时，系统首先通过用户选中的数据集从数据库中获取该数据源最近50个数据，然后判断相应的趋势预测模型对象是否在Redis中，如果在则直接加载并使用，如果不在Redis中则通过读取文件的方式从磁盘中加载至内存并缓存至Redis中。获取模型后进行趋势预测，每次预测使用最新50个时间点的值预测未来一个时间点的值，然后将最新预测的值作为已知值继续预测，直到成功预测未来30个时间点的值停止，最后将预测数据存储至数据库并通过折线图展示该数据未来趋势走向。

趋势预测模块中可以有多个趋势预测模型，每个模型对应着不同的监控项，模型对象间互不关联、相互独立。用户在对模型进行训练的时候需要选择据集的，这确定了模型与数据集之间的一一对应关系，因此在进行趋势预测时只需选择数据集就可以对该数据进行趋势预测。

3.7.2　类图设计

趋势预测模块类图设计如图3.16所示，LSTM是趋势预测模型类，可以存储趋势预测模型的参数信息和方法，每次训练和重置模型时会初始化一个LSTM实例。TrainLSTMModel类主要负责训练和重置趋势预测模型。RedisService负责对Redis数据库操作，包括缓存模型名称和模型对象并设置过期时间。LSTMModelInfo负责趋势预测模型详细信息管理功能，用户可以查看每个趋势预测模型的详细信息并决定是否重置模型。PersistenceService负责趋势预测模型对象的读写操作，当模型对象训练或重置成功时，需要将模型对象持久化，当使用模型对象时，需要将模型对象从磁盘加载至内存中。MySQLService负责趋势预测模块与数据库交互功能。PredictService负责对数据进行趋势预测，当用户选中数据集进行趋势预测时，PredictService通过MySQLService获取该数据集最近数据，通过PersistenceService从磁盘或者通过RedisService从Redis中读取相应的模型对象，然后进行预测，并将预测的趋势走向通过折线图展示给用户。上述各类共同组成趋势预测模块，协作完成训练模型、重置模型、查看趋势预测模型信息和数据趋势预测功能。

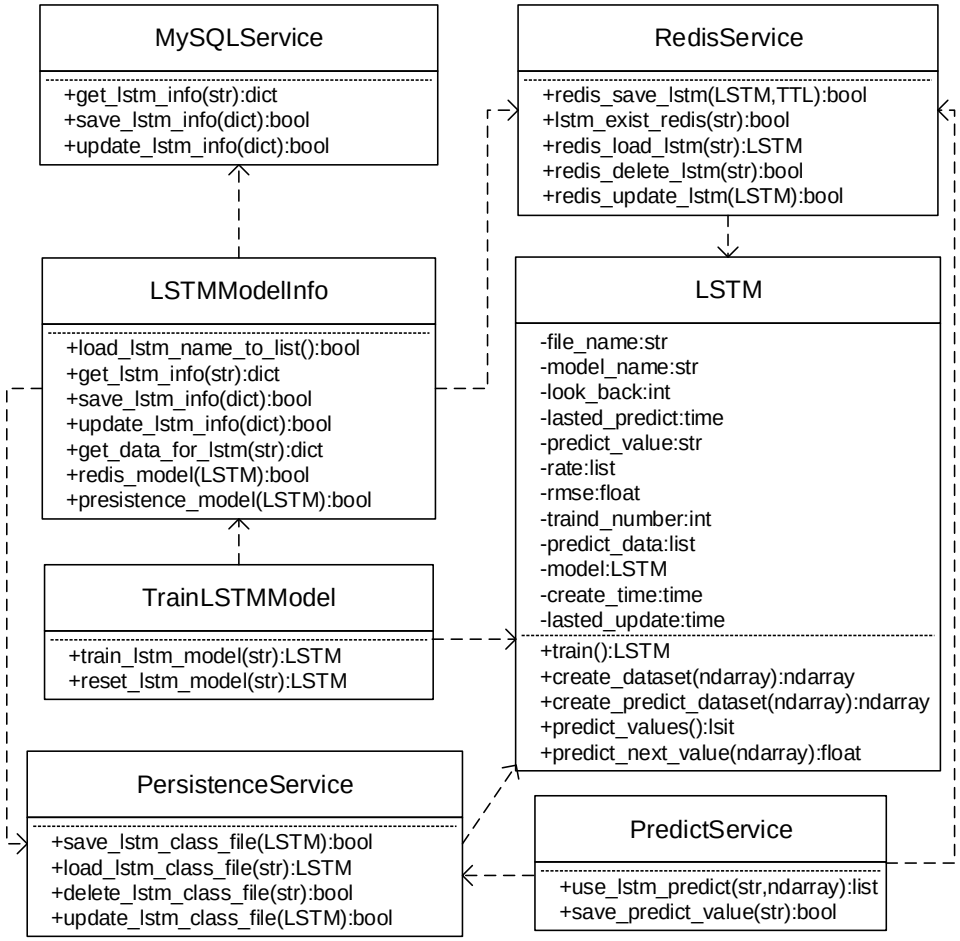

图 3.16: 趋势预测模块类图

3.7.3　趋势预测流程设计

趋势预测流程图如图3.17所示，由于趋势预测模型每次只能预测未来一个值，但是仅预测一个值是不够的，如果采集数据频率是60秒采集一次，只预测一个值也就只能预测未来一分钟趋势走向。当出现资源不足的情况时，运维人员难以在一分钟之内完成资源调度和处理从而避免问题出现。所以系统采用预测未来30个值的方法让用户有充足的时间处理可能的资源瓶颈和问题，预测时首先根据用户选择的数据集获取最新50条数据，并从Redis或磁盘中获取相应的趋势预测模型对象并预测，每次预测完成后将最新预测的值最为已知值继续预测，直到预测完第30个值停止，所有值预测完成后存储至数据库中。

趋势预测模型训练、重置、读取和持久化流程和异常检测流程一样，已在图3.14中给出了详细的解释，这里不再累述。

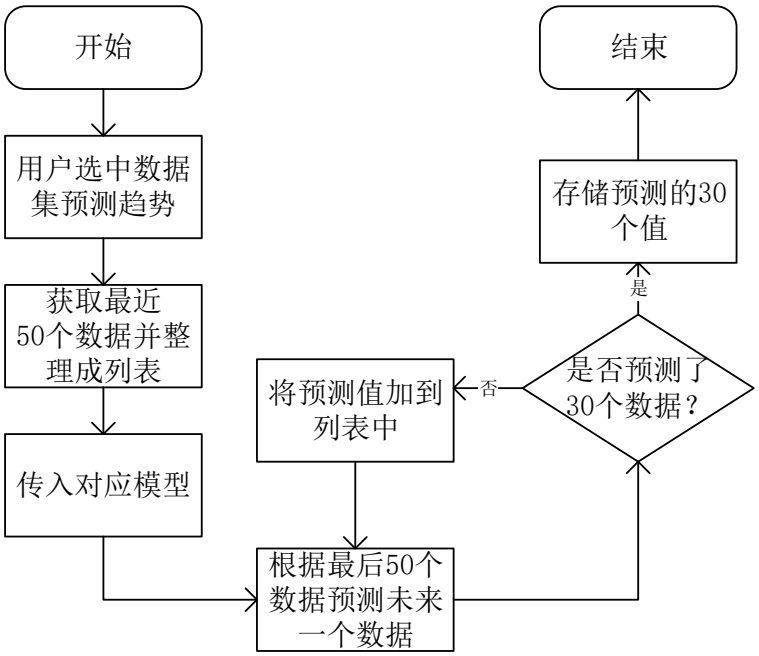

图 3.17: 趋势预测流程图

3.8　数据库设计

数据库主要存储异常检测与趋势预测模型使用的数据、模型详细信息和异常数据信息。数据库需要存储每个数据集的数据，然而用户上传的数据集可以是单指标或多指标的集合，这导致每个数据集结构都不相同，所以需要使用不同表结构存储不同数据集，因此在上传数据集时数据处理模块会解析数据集文件，根据数据集文件第一行标识创建数据库表并将每条数据存储到数据库表中。数据库需要存储每个模型的详细信息，模型信息可分为异常检测模型信息数据表和趋势预测模型信息数据表。数据库也需要存储每个异常信息以便用户集中查看和管理。因此数据库中有三类表。

第一类是用于存储监控项数据的数据表。此类数据来源有两种：一种是用户上传的数据集，系统首先根据数据集的标识ID判断该数据库表是否已存在，如果不存在则建立一张新表，然后使用数据处理模块中孤立森林对数据集进行数据标注预处理并存储至数据库表中；另一种是异常检测模块判断实时数据是否异常后，将数据与标签存储至相应表中。

第二类是模型详细信息表，分为异常检测模型信息表和趋势预测模型信息表。相同类型的模型对象的结构是相同的，所以可以使用同一张表存储。当用户训练或重置模型对象后会生成模型对象的详细信息，系统会将模型对象信息存储至相应表中。

第三类是异常信息表，异常检测模块对实时数据进行判断，如果是异常数据则存储至异常信息表中，方便用户集中查看和管理异常信息。

数据集表、异常信息表和文件名表结构和关系如图3.18所示，数据集表用于记录每个数据集的数据信息，异常信息表用于记录被监控系统的异常信息，文件名表记录数据集名称和唯一标识ID的对应关系。

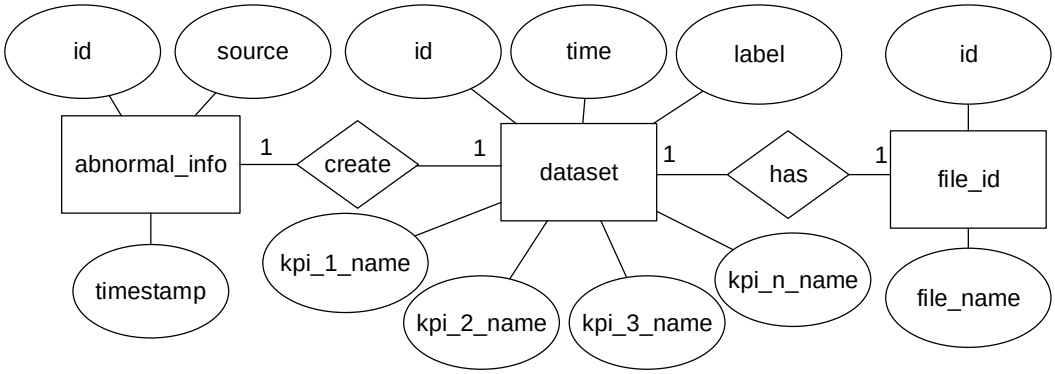

图 3.18: 数据集实体关系图

表3.9是dataset实体关系图的字段说明，主要用于存储数据集信息。其中id是数据集唯一标识，系统通过id判断数据来源。time是产生数据时的时间戳。kpi_n_name表示监控项值，每条数据可能有多个监控项值。label是数据标签，用于表示该条数据是否异常。

表 3.9: dataset表

字段名	数据类型	说明
id	BIGINT	主键，自增，非空。以自增id作为主键而不是时间，避免因为网络拥塞造成时间重合而插入冲突
time	TIMESTAMP	数据产生时间，用于记录数据时间戳
kpi_1_name kpi_n_name	FLOAT	监控项数据组合，数据集由多少监控项数据组成未知，所以表字段由数据集文件第一行字段标识自动生成，每个字段都是float类型
label	FLOAT	标签：1表示异常，0表示正常 数据集数据由孤立森林算法初步生成，后面采集的数据由相应的异常检测模型判断打标后存储

表3.10为file_id实体关系图的字段说明，用于记录数据集名和标识的对应关系，其中file_name表示上传数据集的文件名，id表示数据集的唯一标识UUID。

表 3.10: file_id表

字段名	数据类型	说明
file_name	CHAR(255)	主键，非空。文件上传时的文件名称，方便用户直观查看数据表信息
id	CHAR(255)	非空，数据源唯一标识UUID

表3.11为abnormal_info实体关系图的字段说明，当异常检测模块检测实时数据发现异常后，系统将实时数据的时间和数据来源存储至abnormal_info表中，方便用户集中查看系统异常信息，其中id是主键，time表示异常数据产生时间，source表示异常数据来源。

表 3.11: abnormal_info表

字段名	数据类型	说明
id	BIGINT	主键，自增，非空。以自增id作为主键而不是时间，避免同一时刻多个数据源异常导致插入失败
time	TIMESTAMP	异常数据产生时间
source	CHAR(255)	异常数据来源

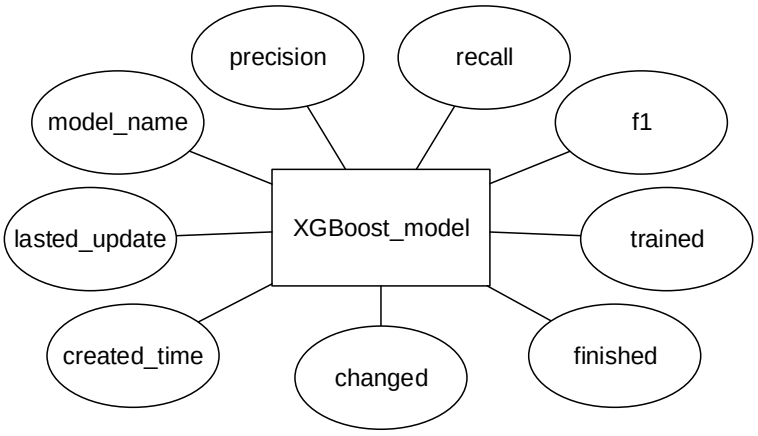

图 3.19: 异常检测模型信息实体关系图

异常检测模型信息表ER图如图3.19所示，其中，model_name表示模型名称，precision表示模型精确率，recall表示模型召回率，f1表示精确率和召回率调和平均值，trained表示模型训练数据量，finished表示模型是否已经完成训练，changed表示数据集标签是否已更改，created_time表示模型创建时间，lasted_update表示模型最后更新时间。所有的异常检测模型详细信息存储在同一张表中，如表3.12所示。

表 3.12: XGBoost_model 表

字段名	数据类型	说明
model_name	CHAR(255)	主键，非空。模型名称以数据集的唯一标识命名，同时也是该数据集的表名称
precision	FLOAT	用于描述模型精确率，即预测为正样本中有多少是真正的正样本，范围为0-1，精确率越高模型效果越好
recall	FLOAT	用于描述模型的召回率，即所有的正样本中有多少被预测到了，范围为0-1，召回率越高模型效果越好
f1	FLOAT	精确率和召回率调和平均，精确率和召回率不会同时很高，二者有一个平衡的过程，而f1就是用来衡量二者综合水平的，范围为0-1，越高模型效果越好
trained	BIGINT	用于说明模型已经训练过的数据量
finished	TINYINT	用于表示该模型是否完成训练：0 表示已经训练完成，1 表示训练未完成
changed	TINYINT	用于说明数据集是否发生变更，数据集如果被用户进行标签更改，则说明模型需要重新训练以得到更为准确的模型。0表示未更改，1 表示已更改
created_time	TIMESTAMP	用于表示模型创建的时间
lasted_update	TIMESTAMP	用于表示模型最后修改时间

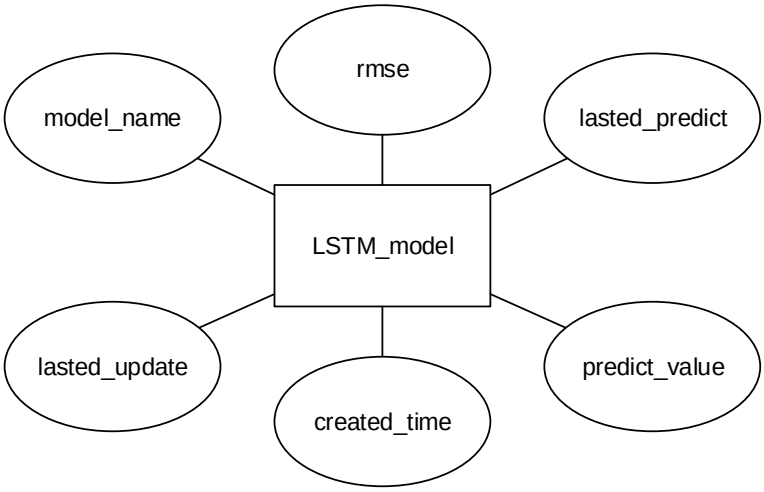

图 3.20: 趋势预测模型信息实体关系图

趋势预测模型信息表ER图如图3.20所示，其中，model_name表示模型名称，rmse表示模型准确率，lasted_predict表示模型最后预测时间，predict_value表示模型最后预测的30个值，created_time表示模型创建时间，lasted_update表示模型最后更新时间。所有趋势预测模型详细信息存储在同一张表，如表3.13所示。

表 3.13: LSTM_model表

字段名	数据类型	说明
model_name	CHAR(255)	主键，非空。模型名以数据集的唯一标识命名，同时也是该数据集的表名称
rmse	FLOAT	用于描述模型的准确率，是模型预测值与真实值之间的均方根误差，越小越模型效果越好
lasted_predict	TIMESTAMP	用于存储模型最后预测的时间
predict_value	VARCHAR	用于存储模型最后一次预测的未来30个时间点的值
created_time	TIMESTAMP	用于表示模型创建的时间
lasted_update	TIMESTAMP	用于表示模型最后修改时间

3.9　本章小结

本章主要说明了基于XGBoost和LSTM的智能监控系统的需求分析和系统设计。首先分析了项目功能性需求和非功能性需求，根据需求分析得到整个系统的总体结构和四个功能模块，并根据4+1视图详细介绍了系统总体设计。然后详细介绍了监控模块、数据处理模块、异常检测模块和趋势预测模块的架构设计、类图设计和流程设计。最后介绍了系统数据库设计，详细介绍了每张数据库表的结构和字段含义。

第四章　智能监控系统的实现和测试

本章在第三章系统设计的基础上，实现了系统的四个功能模块，主要从模块功能顺序图和关键代码介绍了模块的实现，然后通过功能测试和性能测试验证了系统的功能完整性和性能，最后通过效果评估实验证明系统的必要性。

4.1　监控模块的实现

监控模块主要实现两个功能点，一个是多种数据监控项的采集和存储，另一个是异常报警通知用户。Zabbix监控模块是本系统的基础，负责采集服务器信息和应用信息等数据作为数据支撑。本系统采用Client/Server架构作为监控模块架构，使用Zabbix Agent、ICMP、自定义脚本和自定义计算公式的方式采集数据，其中Zabbix Agent负责采集服务器基础信息，ICMP负责采集网络信息，自定义脚本负责采集应用等信息，自定义计算公式负责获取颗粒度更细的数据，监控模块服务器配置如表4.1所示。

表 4.1: 监控模块搭建环境表

CPU	Intel(R) Xeon(R) CPU E52682 v4 @ 2.50GHz × 2
内存	8G
磁盘	100G
操作系统	Ubuntu16.04
数据库	MySQL 5.7.25
Zabbix Server	Version 3.2.11
Zabbix Agent	Version 2.4.7

4.1.1　监控数据采集的实现

系统采集CPU使用信息、内存使用信息、磁盘使用信息、网络带宽信息、网站应用信息、MySQL信息、Docker容器信息、线程进程信息和文件信息等。表4.2和表4.3分别展示部分CPU监控信息和部分网站应用监控信息。

表 4.2: CPU监控信息表

监控项名称	功能	返回值
CPU user time	CPU在非用户进程上使用时间	浮点型
CPU iowait time	CPU等待I/O完成时间	浮点型
CPU idle time	CPU空闲时间	浮点型
CPU使用百分比	CPU运行时使用百分比	浮点型

服务器CPU监控数据如表4.2 所示，通过编写脚本和使用Agent监控服务器CPU使用的详细信息，系统监控CPU在非用户进程使用时间、CPU等待I/O完成时间、CPU空闲时间、CPU使用百分比等数据，表中仅显示部分数据信息。

表 4.3: 网站应用监控信息表

监控项名称	功能	返回值
网站状态	网站登录状态（能否访问和登录）	整型
网站在线用户数	网站实时在线人数	整型
网站用户数量	网站所有用户数量	整型
每日用户注册增加量	网站每日注册用户数量	整型
每日用户登录量	网站每日用户登录总数量	整型

应用信息监控表如表4.3所示，主要使用自定义Python脚本采集应用信息。本系统采集网站运行状态、网站在线用户数、网站所有用户数量、网站每日新注册用户数量、网站每日用户登录总数等信息，表中仅显示部分数据信息。网站应用监控项是和被监控系统是紧密相关的。

系统采集的监控项非常多，这里只展示部分监控项。监控项反应出来的数据特征是本系统需要捕获的重点，每个监控项的具体采集过程如下所示：

（1）在监控服务器端和客户端分别安装Zabbix Server 3.2.11版本和Zabbix Agent 2.4.7版本，客户端负责采集各自的监控项信息，服务器端负责汇总。

（2）修改Server端配置/etc/zabbix/zabbix_agentd.conf中ServerActive=“Server端IP”，Hostname=Zabbix Server，然后重启Server服务，这里“Server端IP”是指Zabbix Server服务器的IP地址。

（3）设置Agent端数据采集信息配置，激活Agent采集配置，并通过配置模板的方式采集基础信息，例如内存、CPU和网络等。

（4）自定义编写Python脚本监控MySQL、Docker和网站应用等信息，通过HTTP请求与网站信息交互获取信息。

（5）将自定义的脚本存储在Agent端的/usr/local/etc/zabbix_script目录下。

（6）修改Zabbix Agent端/etc/zabbix/zabbix_agentd.conf配置，在文件中添加如下配置：UserParameter=“脚本名称”，python “可执行文件”，其中脚本名称是自定义Python脚本名称，可执行文件是Python文件路径，修改并保存成功后重新启动Agent。

（7）在Server端添加第4步编写的自定义脚本，选择数据格式为数值，最后根据不同监控项设定采集频率，例如CPU使用率采集频率为60秒，磁盘使用率采集频率为30分钟等。

监控模块通过上述配置完成信息采集功能，监控模块通过编写获取监控数据的脚本并在不同Agent服务器完成上述配置过程，就可以采集多个Agent服务器的监控指标。

4.1.2　异常数据报警的实现

为了验证本系统的应用能否提高异常报警的准确率，系统既需要使用异常检测模型判断异常并报警，也需要设置固定阈值报警。通过两种报警方式准确率的对比得出系统应用的必要性和异常检测的准确性，所以系统采用传统的固定阈值报警和自定义脚本报警两种策略，其中，自定义脚本报警用于接收来自异常检测模型发出的报警，并通过微信企业号和邮件发送给相关人员。

监控模块的报警媒介为微信企业号和邮件。配置邮件报警与配置自定义监控项过程类似，这里不在累述。配置微信企业号报警需要以下步骤：

（1）申请一个微信企业号，创建一个通讯录，取名为“开发组”，然后将需要了解运维状态的人员通过微信添加至该组。

（2）在微信企业号中创建一个应用，取名为“智能监控系统”，选择部门为第一步设置的开发组，记录该应用AgentId和Secret号码，后续步骤会使用。

（3）编写用于微信企业号报警的Python脚本，脚本作用是当应用获取到异常信息后及时发送给与该应用相关的部门人员，脚本中参数设置为第2步获取的AgentID和Secret。当脚本被执行时会根据AgentID和Secret找到微信企业号的应用和部门，然后将异常信息发送给该应用。

（4）在Server端存储Python脚本并设置脚本权限，本系统中脚本文件存储在/usr/lib/zabbix/alertscripts路径下。

（5）修改Zabbix Server端配置文件/etc/zabbix/zabbix_server.conf，设置Python报警脚本路径AlertScriptsPath=/usr/lib/zabbix/alertscripts，当有异常信息时通过该脚本将信息发送给微信企业号。

（6）点击Zabbix Web界面“管理”菜单栏，然后点击“报警媒介类型”子菜单，选择“创建媒介类型”后进入编辑界面，类型选择为脚本，然后自定义异常报警信息格式和内容。

（7）当Zabbix监控项触发报警时，Python报警脚本会捕获异常信息，通过微信企业号发送给分组成员，关注此企业号的用户可以及时接收报警信息。

上述配置完成后，当系统实时数据与设定的阈值不符或者异常检测模块判定实时数据为异常后，系统会通过微信和邮件报警。由于两种报警策略脚本不同，可以通过报警内容判定是阈值不符合报警或异常检测模块报警，由此统计两种策略的准确率并对比效果。

本系统采用表4.4所示的固定阈值为报警阈值，系统有大量阈值需要设定，表中只罗列几个常用报警阈值的设定。如表4.4所示，当服务器出现CPU实时使用率超过95%、内存实时使用率超过95%、网络状态码不是200、网络丢包率超过2%等情况时会通过阈值报警。

表 4.4: 阈值报警表

监控项名称	阈值设定	说明
CPU使用率	95%	系统实时CPU 使用率超过95%则报警
内存使用率	95%	系统实时内存使用率超过95%则报警
磁盘使用率	95%	系统实时磁盘使用率超过95%则报警
网站状态码	200	网站状态码如果不是200就说明网站有异常则报警
网络丢包率	2%	系统网络丢包率超过2%则报警

4.2　数据处理模块的实现

4.2.1　数据标注顺序图

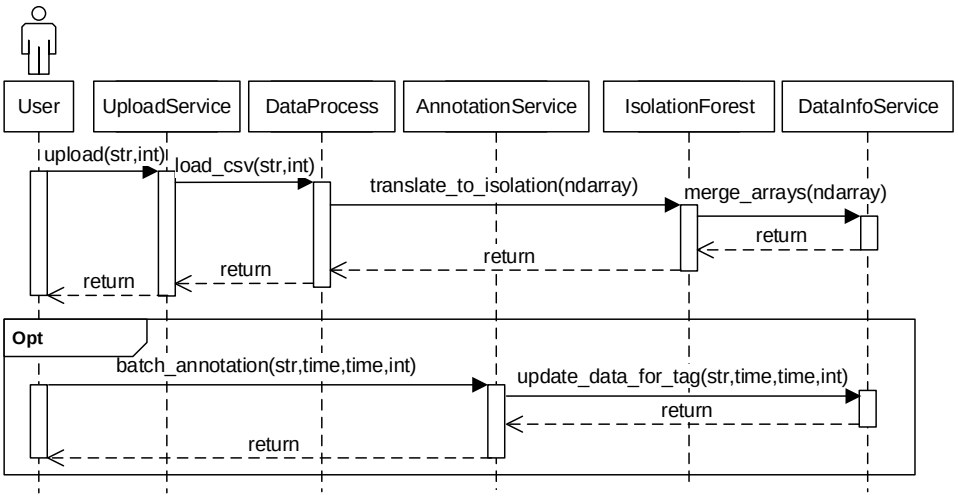

图 4.1: 数据标注顺序图

数据标注如图4.1所示，系统提供两种数据标注的方式：一种是在用户上传数据集的时候，系统会利用孤立森林进行数据标注预处理；另一种是用户通过前端界面批量选择数据进行数据标注。

用户上传数据集时，UploadService接收到用户请求后对数据集进行解析，然后DataProcess对数据格式进行处理并传给IsolationForest孤立森林进行数据标注预处理，得到与数据集中数据一一对应的标签，最后将数据和标签传递给DataInfoService存储至数据库。

用户也可以按数据集名称、时间区间和标签多条件查看数据，并通过多选、单选和全选的方式批量选择数据，然后将这些数据标签重置为正标签或负标签。AnnotationService接收用户选中的数据和标签后，将数据传递给DataInfoService进行标签更新。

4.2.2　关键代码

```
def data_annotation_with_isolate(file_name, abnormal_rate):
    cases = load_csv(file_name)   # file_name 是文件路径名
    title = file_name.split("/")[-1]   # 获取文件名
    isolate_case = Isolate(file_name, cases, rate = abnormal_rate)
    np_array = isolate_case.merge_arrays()
    table_name = np_array[1, 0]
    db = connectdb()
    if not query_table(db, table_name):
        create_table(db, np_array[0], table_name)
    # 插入数据，表名为 uuid
    if insert_train_datas(db, table_name, np_array[1:]):
        # 数据集列表存储表名（redis 存储），断电就清空
        redis_conn = get_redis_connection("default")
        redis_conn.sadd('data_set_name', title)
        # 存储数据集表名（磁盘存储），断电可恢复
        save_dataset_name_to_file(title)
        # 存储文件与 UUID 对应关系到 file2uuid 表中
        insert_file2uuid(title, table_name)
        return True
    return False
```

图 4.2: 孤立森林数据标注预处理代码

图4.2是孤立森林数据标注预处理并存储的代码，主要负责解析数据集并使用孤立森林进行数据标注预处理，该函数接收file_name和abnormal_rate两个参数，其中file_name表示数据集名称，abnormal_rate表示数据集异常比例。函数执行时，模块通过file_name获取文件并解析内容，将内容格式化处理后传给孤立森林算法进行数据标注预处理并返回处理结果。存储结果时，模块首先判断数据库中是否存在该数据集表，如果不存在则需要将文件第一行属性作为数据库表字段创建新的数据库表，然后将结果数据写入数据库表中，如果数据表已经创建，则直接在该表中添加结果数据。最后将数据集名称与数据集唯一标识UUID的对应关系存入file_id表中，将数据集名称缓存至Redis和磁盘文件中，当用户重复上传数据集时，系统可以快速检索Redis判断数据集是否已上传。

4.3　异常检测模块的实现

4.3.1　异常检测顺序图

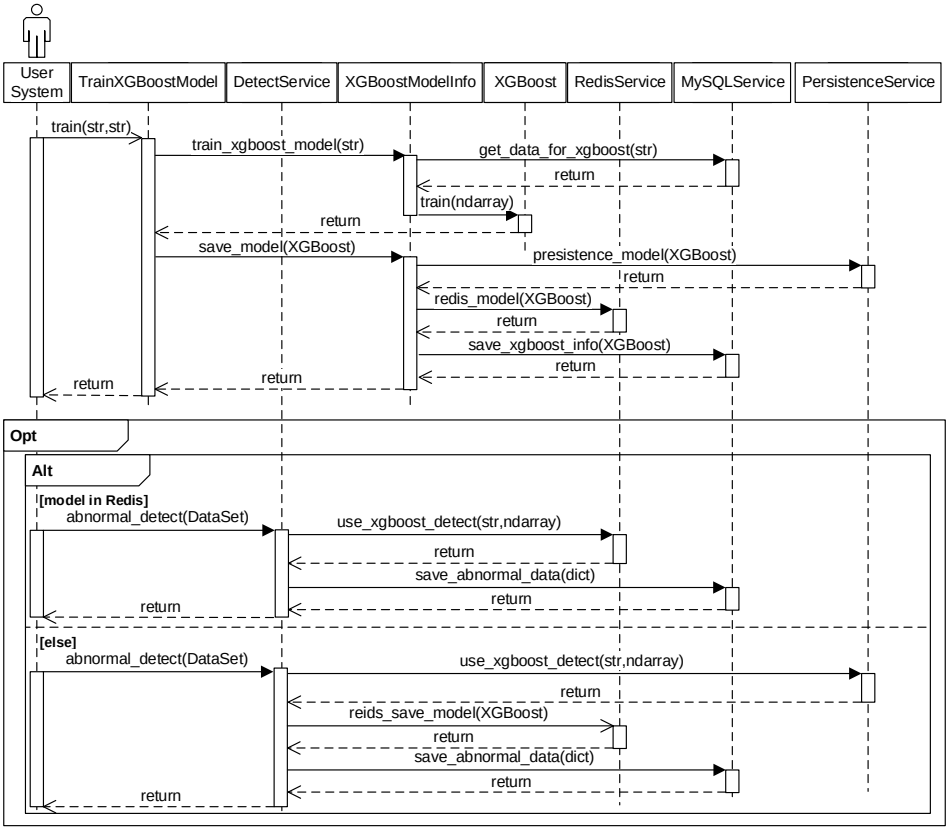

图 4.3: 异常检测顺序图

异常检测顺序图如图4.3所示，主要描述了训练异常检测模型和实时数据异常检测两个流程。

训练异常检测模型时，TrainXGBoostModel用于接收用户训练异常检测模型的异步请求，然后将数据集传给XGBoostModelInfo，XGBoostModelInfo从数据库中获取训练该模型需要的数据，然后将数据传递至XGBoost模型中进行训练。训练完成后，XGBoostModelInfo调用PresistenceService将模型对象持久化至磁盘，调用RedisService将模型对象缓存至Redis并设置过期时间，调用MySQLService存储模型对象信息，最后提示用户模型训练成功。

对实时数据进行异常检测时，DetectServer接收实时数据后对数据进行解析，判断数据来源并获取相应的异常检测模型，获取模型对象时存在两种情况：一种是模型对象在Redis中，DetectServer通过RedisService从Redis中获取与实时数据相对应的模型对象；另一种情况是模型对象不在Redis中，DetectService通

过PresistenceService从磁盘中获取模型对象，将模型对象缓存至Redis并设置过期时间，然后DetectServer使用异常检测模型对数据进行异常检测，并通过MySQLService将数据和标签存入相应的数据表中。

4.3.2　XGBoost模型参数设置

异常检测模块使用的XGBoost算法模型参数多，选择合适的参数可以提高数据异常检测的准确率，因此本小节利用实验确定XGBoost算法模型最优参数，提高异常检测准确率。XGBoost模型参数分为三类：第一是通用参数，用于宏观函数控制；第二是Booster参数，用于控制每一颗树构造过程；第三是学习目标参数，用于控制训练模型效果。在通用参数中，由于本系统中XGBoost作用是分类，所以通用参数booster选择gbtree树结构对数据分类。训练模型数据集选取某服务器2018年12月15日至2019年2月27日74天共107717条内存使用率和CPU使用率的数据。使用孤立森林对数据进行数据标注预处理后，选取数据集中70%数据共计75401条数据为训练集，30%的数据共计32316条数据作为测试集，训练过程中不改变数据标签以保证数据的一致性。

表 4.5: XGBoost算法初始参数值

参数名	参数值	参数名	参数值
booster	gbtree	**verbosity**	0
objective	binary:logistic	**max_depth**	10
eta	0.7	**subsample**	0.7
evals	['error', 'auc', 'rmse']	**gamma**	0
lambda	10	**min_child_weight**	2

在进行实验前，需要初步确定参数，并在实验过程中逐个调整为最优参数。首次训练算法模型的参数设定如表4.5所示，其中，booster是助推器，本文选择最合适的gbtree解决分类问题。verbosity用于控制程序运行消息详细程度。objective选用二元分类的逻辑回归，将数据分为正类和负类两种。max_depth是树的最大深度。subsample是训练实例的子样本比例，每次选取训练集的一定比例数据来训练，防止过拟合。evals用于评估模型效果，min_child_weight是叶子最小权重，min_child_weight、eta、gamma和lambda用于控制树分裂过程，防止过拟合。参数调优具体实验步骤如下：

第一步，确定最优迭代次数。迭代次数是指在训练过程中创建多少棵树进行训练，树的数量太少会造成准确率低，数量太多会造成过拟合。根据经验，首先训练选择迭代次数为10000次，选择“early_stopping_rounds”=1000，该参

数的作用是如果测试集的错误率在连续1000次迭代中都没有下降则停止训练，此时就能确定最优迭代次数。迭代次数实验如图4.4所示，从图中可以看出，算法模型从154到1154次迭代中，测试集错误率都没有降低，由此可确定算法模型最优迭代次数为154。

```
[1149]  train-error:0.008355    test-error:0.035248
[00:02:44] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74
[1150]  train-error:0.008316    test-error:0.035186
[00:02:44] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74
[1151]  train-error:0.008236    test-error:0.035155
[00:02:44] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74
[1152]  train-error:0.008223    test-error:0.035155
[00:02:44] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74
[1153]  train-error:0.008223    test-error:0.035155
[00:02:44] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74
[1154]  train-error:0.008183    test-error:0.035186
Stopping. Best iteration:
[154]   train-error:0.025623    test-error:0.032215
```

图 4.4: 迭代次数实验图

第二步，确定每棵梯度提升树的最大深度max_depth。如果树的深度太小会造成分类不完全，准确率不高；如果深度太大会造成过拟合。实验时首先选择其他参数为默认值，通过对比不同的max_depth取值对模型准确率的影响，得出树深度最优取值。实验中选择本系统比较关注的精确率、召回率和二者调和平均值f1作为判断模型准确率依据。根据经验，max_depth一般取值为[3,10]，分别使用不同max_depth进行实验，实验结果如表4.6所示，从表中可以看出，当最大深度值为7时模型效果最好。

表 4.6: max_depth参数实验

max_depth	3	4	5	6	7	8	9	10
精确率	0.9154	0.9199	0.9089	0.8963	0.9041	0.8901	0.8821	0.8601
召回率	0.8313	0.8413	0.8503	0.8878	0.8830	0.8843	0.8835	0.8985
f1值	0.8713	0.8788	0.8787	0.8921	0.8934	0.8848	0.8829	0.8789

第三步，确定步长eta参数。eta参数用于缩小数据特征权重以防止训练过度拟合。根据经验，eta一般取值为[0.01,0.3]。实验时固定其他参数（此时max_depth使用最优参数7），不断更改eta值进行实验，通过对比精确率、召回率和f1值得出最优eta参数。如表4.7所示，实验证明最优eta参数值为0.20。

表 4.7: eta参数实验

eta	0.01	0.025	0.05	0.10	0.15	0.20	0.25	0.30
精确率	0.9290	0.9082	0.8976	0.8959	0.8932	0.8993	0.8830	0.8624
召回率	0.8410	0.8443	0.8799	0.8902	0.8853	0.8895	0.8862	0.8862
f1值	0.8829	0.8751	0.8887	0.8931	0.8893	0.8943	0.8846	0.8742

重复上述实验过程，依次确定其他参数，这里不再累述。最终确定最优参数如表4.8所示，确定最优参数后将其应用至其他数据集中，发现模型效果相近，所以本系统异常检测模型采用表4.8中参数作为模型最优参数。

表 4.8: XGBoost算法最优参数值

参数名	参数值	参数名	参数值
booster	gbtree	**verbosity**	0
objective	binary:logistic	**max_depth**	7
eta	0.2	**subsample**	0.6
evals	[’error’, ’auc’, ’rmse’]	**gamma**	1
lambda	9	**min_child_weight**	2

4.3.3 关键代码

```
def init_model(self):
    datas=load_data_for_xgboost_from_mysql(self.name) #从数据库获取数据
    np.random.shuffle(datas) #按行打乱顺序，然后从中选择训练集和测试集
    rate=[7,3] # 训练集和测试集取 7:3
    self.trained_number=len(datas) #训练集总数量
    total_rate=sum(rate) # 总比例，用于取出训练集和测试集
    rate_num1=int(self.trained_number*rate[0]/total_rate)
    dtrain=xgb.DMatrix(datas[0:rate_num1,0:-1].astype(float),
                       label=datas[0:rate_num1,-1].astype(int)) #训练集
    dtest=xgb.DMatrix(datas[rate_num1+1:-1,0:-1].astype(float),
                      label=datas[rate_num1+1:-1,-1].astype(int)) #测试集
    watchlist=[(dtrain,'train'), (dtest,'test')] # 显示训练过程
    bst=xgb.train(self.param,dtrain,self.num_round,watchlist) # 训练模型并验证
    preds=bst.predict(dtest) # 预测测试集数据
    self.precision,self.recall=get_label(dtest.get_label()) #精确率和召回率
    self.f1=self.precision*self.recall*2/float(self.precision+self.recall) #f1 值
    self.lasted_update=time.strftime("%Y-%m-%d %H:%M:%S",time.localtime())
    self.model=bst #更新模型
    self.update_database_model() #更新数据库
```

图 4.5: XGBoost类初始化代码

图4.5是XGBoost类初始化代码，用户每次训练模型时系统会初始化一个XGBoost实例并调用图4.5中方法进行初始化。初始化时，系统首先从数据库中获取对应的数据集数据，处理数据集信息格式并按7:3的比例分为训练集和测试集，然后使用训练集训练模型并使用测试集测试模型的精确率、召回率

和f1值，最后更新模型信息并将模型持久化。在模型训练完成后，用户可以对数据集数据标签进行更正并重新训练模型，以提高模型的精确率和召回率，所以本系统使用时间越长，用户对数据标注越多，算法模型越准确效果越好。

4.4　趋势预测模块的实现

4.4.1　趋势预测顺序图

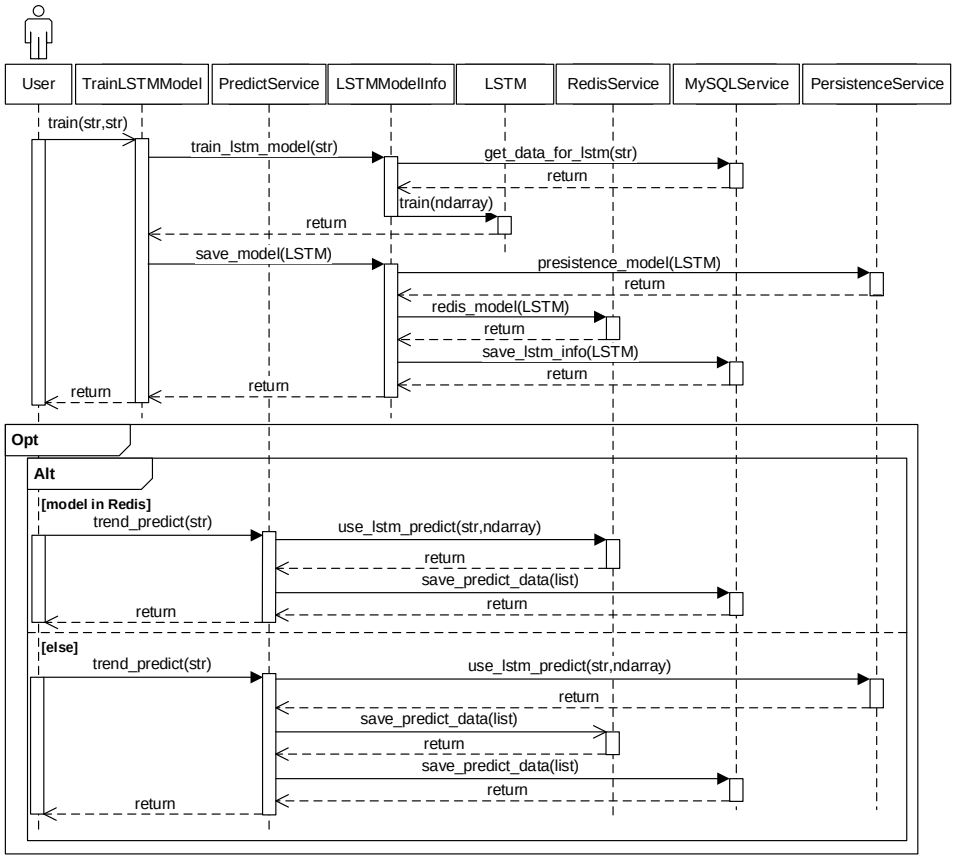

图 4.6: 趋势预测顺序图

趋势预测顺序图如图4.6所示，图中主要描述了训练趋势预测模型和对数据进行趋势预测两个流程。

在训练趋势预测模型流程中，TrainLSTMModel接收用户训练趋势预测模型的异步请求，然后将选中的数据集传给LSTMModelInfo。LSTMModelInfo通过MySQLService获取训练趋势预测模型需要的数据并进行数据格式处理，然后调用LSTM模型进行训练。模型训练完成后调用PersistenceService将模型持久化至磁盘，调用RedisService将模型对象缓存至Redis并设置过期时间，调

用MySQLService将模型信息存储到数据库中。最后将训练结果返回给用户，整个训练过程异步后台进行，不影响用户使用体验。

用户选中数据集进行趋势预测时，系统通过PredictService接收用户请求，然后获取相对应的趋势预测模型对象，获取模型对象的过程和获取异常检测模型对象一样，这里不再累述。获取模型对象后使用模型进行预测，预测时系统会获取用户选中数据集的最近50个时间点的值，以此为基础预测下一个时间点的值，然后不断循环预测，直到预测出未来30个时间点的值时停止。最后将预测的值存入数据库并通过前端ECharts折线图展示趋势走向。

4.4.2　关键代码

```
def predict_values(self):
    # 获取最后 50 个数据作为预测的输入
    data=load_data_for_lstm_from_mysql(self.name, 50)
    data=np.reshape(data, (len(data), 1)) # 数据维度转换
    data=data[-50:, :].tolist() # 数据格式转换
    self.predict_data=data #初始化预测值
    # 循环预测，直到预测出 30 个值为止
    while len(self.predict_data)<self.look_back+self.look_forward:
        tmp=self.predict_next_value(data = self.predict_data)
        self.predict_data.append(tmp)
    self.predict_data=sum(self.predict_data, []) #二维数据转一维数据
    self.predict_data=[round(i, 2) for i in self.predict_data] # 精确到小数点后 2 位
    str_value = self.predict_data[(-1 * self.look_forward):] # 取后三十个值存储
    self.predict_str_value = ','.join(str(e) for e in str_value) # 格式转换
    self.lasted_predict=time.strftime("%Y-%m-%d %H:%M:%S",time.localtime())
    self.update_database_model() # 更新数据库表
    return str_value #返回预测值
```

图 4.7: 数据趋势预测代码

图4.7是数据趋势预测代码，函数predict_values的主要功能是使用趋势预测模型对象预测未来30个时间点的值并存储。用户选择数据集并进行预测时，相应的趋势预测模型调用predict_values函数进行趋势预测，predict_values首先从数据库中获取对应数据集的最新50个数据。经过数据维度和格式转换，predict_values将50个值传入模型进行循环预测，每次调用predict_next_value函数预测未来一个值，并将该预测值作为已知值继续预测，直到预测出未来30个时间点的值时停止。最后将预测值进行格式化处理，存储至数据库中并返回。

4.5　系统测试与运行展示

4.5.1　测试目标

本系统测试主要从功能测试和性能测试两方面进行：功能测试用于验证系统的功能的完整性和正确性；性能测试主要测试系统接口在面对高并发请求时的处理及时性和正确性；测试分为以下两部分：

第一部分是功能测试。这部分主要测试用户上传数据集、批量数据标注、管理数据信息、查看异常数据、查看异常检测和趋势预测模型信息、训练和重置模型、实时数据异常检测和趋势预测等功能是否正常运行。

第二部分是性能测试。这部分主要使用Apache Jmeter性能压测工具对系统主要接口进行压力测试，测试系统接口在面对高并发请求时能否保证请求处理的及时性和正确性。

在测试开始之前，需要将本系统部署到服务器上，然后在真实环境下进行测试。系统部署环境如表4.9所示，系统和环境部署完成后对系统进行测试。

表 4.9: 系统部署环境表

设备与软件	环境部署
系统服务器	四台，分别部署主服务、Redis、Mysql数据库、Zabbix Server
被监控服务器	若干，每台服务器上部署Zabbix Agent并与Zabbix Server关联
Redis服务器	Redis Version 3.2.0
数据库服务器	MySQL Version 5.7.25
主服务器硬件环境	8G内存，100G磁盘，50M带宽
主服务器软件环境	Ubuntu16.04，Python3.6.4，Django2.1.5，Keras2.2.4

4.5.2　功能测试

本小节依据第三章需求分析得到的功能进行用例设计与测试。设计时测试用例要尽量覆盖所有系统功能，本小节通过设计多个测试用例来验证系统功能的完整性和正确性，其中每个测试用例包含测试ID、测试名称、测试功能、测试步骤、预期结果和测试结果。测试选用某台被监控服务器的真实内存使用量数据，下文中使用“A服务器内存使用量”表示。

表4.10是上传数据集测试用例，主要测试系统上传数据集功能是否正常运行。用户首先点击“上传数据集”菜单按钮进入上传数据集界面，此时系统应该跳转至上传数据集页面。然后用户点击“选择文件”按钮，选中数据集文件和该数据集的异常数据比例后点击“上传”按钮，系统此时接收数据集，后台调用孤立森林算法模型对数据集进行数据标注预处理，然后将数据和标签存储至数据库并提示用户上传成功。

表 4.10: 上传数据集测试用例

测试ID	**TC1**
测试名称	上传数据集
测试功能	用户通过前端界面指定异常比例并上传数据集
测试步骤	1.点击“上传数据集”菜单按钮 2.点击“选择文件”按钮后，选择待上传的数据集为“A服务器内存使用量”，并选择该数据集的异常数据比例（可不选，默认为0.1） 3.点击“上传”按钮
预期结果	1.页面跳转至“上传数据集页面” 2.系统弹出文件选择对话框，用户选中后界面显示用户选中的文件名称为“A服务器内存使用量” 3.系统显示数据集“A服务器内存使用量”上传成功，并提示是否“立即训练模型”或者“继续上传数据集”
测试结果	通过

表 4.11: 查看数据集信息测试用例

测试ID	**TC2**
测试名称	查看数据集信息
测试功能	用户通过多条件查看数据集信息，并可以对数据进行检索和排序
测试步骤	1.点击“数据标注”菜单按钮 2.选择数据集名称为“A服务器内存使用量”、开始时间为“2019/02/25 14:10”、结束时间为“2019/02/25 14:40”、标签为“所有”，然后点击“查询”按钮 3.在界面左上角点击下拉框，选择表格显示行数为“50”行 4.点击界面中表格的“时间”栏 5.点击界面中表格的“标签”栏 6.在界面右上角“Search”文本框中输入“2019/02/25 14:10” 7.点击界面右下角“Next”按钮 8.点击界面右下角“Previous”按钮
预期结果	1.页面跳转至“数据标注”页面 2.系统按照用户选择的条件获取数据并通过表格的形式展示，每页表格只显示10行数据信息，每行分别显示数据的获取时间、值和标签 3.每个页面表格变成同时显示50行数据 4.表格按照“时间”升序排列数据 5.表格按照“标签”升序排列数据 6.表格只显示时间是“2019/02/25 14:10”的一行数据 7.表格显示下一页的数据 8.表格返回上一页，显示之前一页数据
测试结果	通过

表4.11是查看数据集信息测试用例，主要验证系统能否多条件查看数据信息，并实现排序、搜索和翻页等功能。用户首先点击”数据标注“菜单按钮，

此时系统页面跳转至”数据标注“页面。然后用户选择数据集名称、时间区间和标签并点击“查询”按钮，此时系统通过表格按用户选定条件显示数据，用户可以对数据进行排序、搜索、翻页和更改每页显示数据量。

表4.12是批量标注数据集测试用例，主要用于验证系统数据批量标注功能。用户首先点击“数据标注”菜单按钮，此时页面跳转至“数据标注”页面。用户通过多选、单选和全选等方式选中要进行标注的数据并点击“标签置为0”或者“标签置为1”按钮，系统接收用户请求并更新选中数据的标签。

表 4.12: 批量标注数据集测试用例

测试ID	**TC3**
测试名称	批量数据标注数据集
测试功能	用户通过单选、多选和全选的方式选中数据并进行批量数据标注
测试步骤	1.点击“数据标注”菜单按钮 2.选择测试集名称为“A服务器内存使用量”、开始时间为“2019/02/25 14:20”、结束时间为“2019/02/25 14:40”、标签为“1”，然后点击“查询”按钮 3.点击表格中第一行数据末尾“标记”按钮 4.点击选中表格第2、3、4行开头复选框 5.点击“Next”翻页 6.点击选中表头复选框 7.点击“标签置为0”按钮
预期结果	1.页面跳转至“数据标注”页面 2.系统前端界面展示“A服务器内存使用量”数据集在“2019/02/25 14:20”至“2019/02/25 14:40”时间内的所有标签为“1”的数据 3.第一行数据标签被重置为“0” 4.第一页第2、3、4行被选中 5.页面跳转下一页，表格中内容切换到下一页 6.页面所有数据被选中 7.表格第一页第2、3、4行数据和第二页所有数据标签被更改为“0”
测试结果	通过

表4.13是训练异常检测模型测试用例，主要用于验证训练异常检测模型功能。用户首先进入“异常检测模型信息”界面查看“A服务器内存使用量”异常检测模型是否存在，由于此时该模型还未训练，在“异常检测模型信息”界面没有该模型信息。用户点击“训练模型”菜单按钮，选择“模型类型”为“异常检测”，选中数据集为“A服务器内存使用量”后点击“开始训练”按钮，此时系统应该接受请求，根据模型类型调用XGBoost算法模型异步训练模型，训练完成后将模型信息存入数据库中。用户再次进入“异常检测模型信息”界面就能查看“A服务器内存使用量”模型详细信息。

表 4.13: 训练异常检测模型测试用例

测试ID	TC4
测试名称	训练异常检测模型
测试功能	用户通过界面异步训练异常检测模型
测试步骤	1.点击“异常检测模型信息”菜单按钮 2.点击“训练模型”菜单按钮 3.选择“模型类型”为“异常检测”，数据集为“A 服务器内存使用量” 4.点击“开始训练”按钮 5.系统提示“训练成功”后点击“异常检测模型信息”菜单按钮 6.界面显示表格，点击两次“创建时间”表头
预期结果	1.页面跳转至“异常检测模型信息”页面，所有模型在“数据来源”一栏里没有“A服务器内存使用量”，说明该模型尚未训练 2.页面跳转至“训练模型”页面 3.“模型类型”下拉菜单显示“异常检测”，“数据集”下拉菜单显示“A服务器内存使用量” 4.页面提示“模型后台训练中”，用户可以进行其他操作 5.界面显示所有异常检测模型详细信息 6.所有模型信息按模型创建时间降序排列，第一行数据的“数据来源”是“A服务器内存使用量”，其“精确率”、“准确率”、“f1”值都不为0
测试结果	通过

表 4.14: 训练趋势预测模型测试用例

测试ID	TC5
测试名称	训练趋势预测模型
测试功能	用户通过界面异步训练趋势预测模型
测试步骤	1.点击“趋势预测模型信息”菜单按钮 2.点击“训练模型”菜单按钮 3.选择“模型类型”下拉菜单为“趋势预测”、数据集下拉菜单为“A服务器内存使用量”，然后点击“开始训练”按钮 4.系统提示“训练成功”后点击“趋势预测模型信息”菜单按钮 5.界面显示表格，点击两次“创建时间”表头
预期结果	1.页面跳转至“趋势预测模型信息”页面，所有模型信息在“数据来源”一栏里没有“A服务器内存使用量” 2.页面跳转至“训练模型”页面 3.页面提示“模型后台训练中”，用户可以进行其他操作 4.界面显示所有趋势预测模型详细信息 5.所有模型信息按模型创建时间降序排列，第一行数据的“数据来源”是“A服务器内存使用量”，其“均方根误差”值不为0
测试结果	通过

表4.14是训练趋势预测模型测试用例，主要用于验证训练趋势预测模型功能。验证过程与验证训练异常检测模型相同，这里不再累述。

表4.15是系统实时数据异常检测的测试用例，主要用于验证异常检测模型训练完成后，系统能否对实时数据进行异常检测并通过微信和邮件报警。该测试用例选用Postman来模拟发送请求，分别选用一个正常数据和一个异常数据进行测试，系统只有接收异常实时数据后会通过微信和邮件报警。

表 4.15: 系统实时数据异常检测测试用例

测试ID	**TC6**
测试名称	系统实时数据异常检测
测试功能	对系统实时数据进行异常检测
测试步骤	1.打开Postman软件，URL 填写“http://101.37.78.167:9090/models/sbmit”，选择协议为“POST”请求，“Body”中填写一条“A 服务器内存使用量”的真实数据值：{“host_id”：“26035cd8-eff2-4802-82db-34674afe8fa9”，“time”：“2019-02-25 14:26:26”，“kpi”：“31.16”}，点击“Send”按钮 2.修改第二步中“Body”的“kpi”值为“131.16”并点击“Send”按钮
预期结果	1.系统返回“0”代表该实时数据是正常的 2.系统返回“1”，说明该实时数据是异常的，因为“kpi”值是“131.16”超过内存最大使用量100%，所以是异常数据，微信企业号收到报警消息，邮件收到报警消息
测试结果	通过

表4.16是数据趋势预测的测试用例，主要用于验证系统能否对数据进行趋势预测。用户进入“趋势预测”界面，选中数据集并点击“预测”按钮后，系统接收用户预测请求，根据数据集调用相应的趋势预测模型进行预测，将结果存储至数据库并通过折线图展示预测趋势。

表 4.16: 数据趋势预测测试用例

测试ID	**TC7**
测试名称	数据趋势预测
测试功能	用户通过界面预测某数据未来趋势走向
测试步骤	1.点击“趋势预测”菜单按钮 2.选择数据集为“A服务器内存使用量”并点击“预测”按钮
预期结果	1.页面跳转至“趋势预测”页面 2.页面显示“A服务器内存使用量”未来30个时间点趋势走向折线图
测试结果	通过

表4.17 是查看异常检测模型信息与重置模型测试用例，主要用于验证查看异常检测模型详细信息和重置模型功能是否正常运行。用户进入“异常检测模型信息”界面后，可以通过排序、搜索和翻页等方式查看模型详细信息，可以选中模型信息并点击“重置模型”按钮进行重置。系统接收重置请求后，获取

该数据集的数据重新训练模型，并更新原有模型对象和模型信息。在“异常检测模型信息”界面，列表单元格根据每个模型的“f1”值显示不同颜色。

表 4.17: 查看异常检测模型信息与重置模型测试用例

测试ID	TC8
测试名称	查看异常检测模型信息与重置模型
测试功能	用户通过界面查看异常检测模型信息并重置模型
测试步骤	1.点击“异常检测模型信息”菜单按钮 2.点击表头“f1”值，按升序排列 3.选中“A服务器内存使用量”模型所在行，并点击“重置模型”按钮 4.系统提示“训练成功”后点击“异常检测模型”菜单按钮，查看“A服务器内存使用量”所在行的“f1”值和最后“最后更新”
预期结果	1.页面跳转至“异常检测模型信息”页面 2.表格按“f1”值升序排列，f1值小于0.5时单元格显示为红色，在0.5至0.75区间时单元格为橙色，大于0.75时单元格为绿色 3.系统提示“模型后台训练中” 4.“A服务器内存使用量”一行“f1”值发生变化，“最后更新”为当前时间，说明异常检测模型重新训练成功
测试结果	通过

表4.18是查看趋势预测模型信息与重置模型测试用例，主要用于验证查看趋势预测模型详细信息和重置模型功能是否正常运行，其测试流程与查看异常检测模型信息类似，这里不再累述。本测试用例主要关注列表单元格是否根据模型“均方根误差”显示不同颜色、模型能否重置成功。

表 4.18: 查看趋势预测模型信息与重置模型测试用例

测试ID	TC9
测试名称	查看趋势预测模型信息与重置模型
测试功能	用户通过界面查看趋势预测模型信息并重置模型
测试步骤	1.点击“趋势预测模型信息”菜单按钮 2.点击表头“均方根误差”值，按升序排列 3.点击“A服务器内存使用量”所在行的“重置模型”按钮 4.待系统提示“训练成功”后点击“异常检测模型”，查看“A服务器内存使用量”行的“均方根误差”值和最后“模型最后更新时间”
预期结果	1.页面跳转至“趋势预测模型信息”页面 2.表格按“均方根误差”值升序排列，均方根误差小于100显示绿色，高于100显示橙色表示准确率较低 3.系统提示“模型后台训练中” 4.“A服务器内存使用量”一行“均方根误差”值发生变化，“模型最后更新时间”为当前时间，趋势预测模型重新训练成功
测试结果	通过

表4.19是查看异常信息测试用例，用于验证用户能否在前端集中查看历史异常数据信息。被监控系统产生实时数据后，异常检测模块调用相应异常检测模型对实时数据进行异常检测，如发现异常则将数据集中存储至数据库中，便于用户集中查看异常信息。

表 4.19: 查看异常信息测试用例

测试ID	TC10
测试名称	查看异常信息
测试功能	用户通过界面查看历史异常信息
测试步骤	1.点击“异常信息”菜单按钮 2.选择时间区间为“2019-02-25 14:20”至“2019-02-25 14:30”并点击“查询”按钮
预期结果	1.页面跳转至“异常信息”页面 2.页面显示这段时间区间内的所有异常值
测试结果	通过

4.5.3　性能测试

本节从性能方面对系统进行测试，使用Apache Jmeter性能压测工具实现接口压力测试。Apache Jmeter是一款开源测试工具，专门用于测试Web应用接口性能，可以模拟大量负载访问Web应用。下面测试本系统几个主要接口的性能，分别选取并发数100、200、300和400，测试实时数据异常检测、查看异常检测模型信息、查看趋势预测模型信息、训练模型、查看数据集信息、批量数据标注和趋势预测七个接口。图4.8是1s内并发执行100次接口访问的详细结果。

Label	# 样本	平均值	最小值	最大值	异常 %	吞吐量
实时数据异常检测	100	666	13	1610	0.00%	41.2/sec
查看异常检测模型信息	100	212	11	624	0.00%	38.6/sec
查看趋势预测模型信息	100	203	13	704	0.00%	36.5/sec
训练模型	100	46	4	159	0.00%	35.7/sec
查看数据集信息	100	44	5	166	0.00%	35.2/sec
批量数据标注	100	43	5	158	0.00%	34.8/sec
趋势预测	100	40	6	162	0.00%	34.8/sec
总体	700	179	4	1610	0.00%	238.0/sec

图 4.8: 压力测试并发数为100时结果汇总

从图4.8中可以看出，1s内并发访问七个接口100次的平均响应时间是179ms，最小响应时间4ms，最大响应时间1610ms，异常比例为0，服务器每分钟处理请求数为238条请求/秒。其中，实时数据异常检测接口平均响应时间为666ms，查看异常检测模型信息接口平均响应时间为212ms，查看趋势预测模型信息接口平均响应时间为203ms，训练模型接口平均响应时间为46ms，查看数据集信息

接口平均响应时间为44ms，批量数据标注接口平均响应时间为43ms，趋势预测接口平均响应时间为40ms。

由此方法，另外分别测试1s内并发数为200、300和400时系统性能情况，结果如表4.20所示。从表中可以看出，并发量的增加会导致接口平均响应时间增大。随着并发量的增加，系统平均吞吐量维持在215条请求/秒，异常比例一直为0。当1s内并发访问某接口400次时，接口异常比例为0，平均响应时间为1173ms，此响应时间较短，由此可证明系统某接口在1s内接受400次以下请求时，能保证请求处理的及时性和正确性。

表 4.20: 性能压测结果表

并发量	平均值	最小值	最大值	异常	吞吐量
100	179ms	4ms	1610ms	0%	238.0 条请求/秒
200	524ms	5ms	4531ms	0%	213.7 条请求/秒
300	885ms	5ms	7643ms	0%	206.1 条请求/秒
400	1173ms	5ms	10528ms	0%	202.2 条请求/秒

4.5.4　系统运行展示

系统主要提供上传数据集界面、训练模型界面、异常检测与趋势预测模型详细信息界面、趋势预测界面、数据标注界面和异常信息界面。

上传数据集界面提供给用户指定数据集异常比例并上传数据集，系统接收数据集文件并提示用户上传成功。训练模型界面中，用户可以选择数据集和模型类型并训练模型，系统根据模型类型调用相应的算法模型后台异步进行训练。在异常检测模型详细界面中，用户可以查看所有异常检测模型的信息，并可以通过点击“重置模型”按钮重新训练模型。异常检测模型信息界面如图4.9所示，图中模型信息按“f1”值倒序排列，并通过不同颜色填充单元格提示用户是否应该重置模型，其中红色代表准确率低，橙色代表准确率一般，绿色代表准确率高。趋势预测模型信息界面和异常检测模型界面相似，这里不再累述。

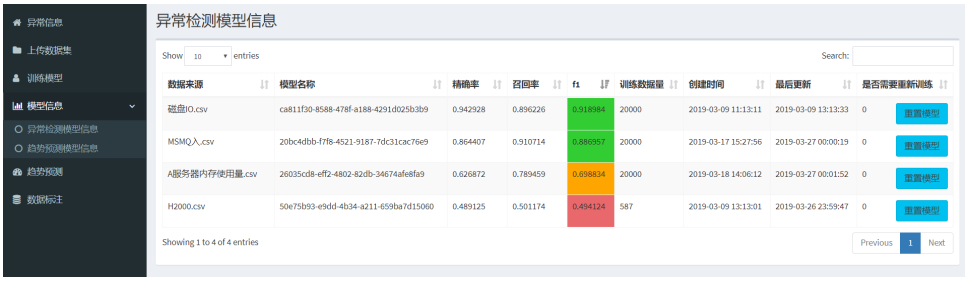

图 4.9: 异常检测模型信息界面

趋势预测界如图4.10所示，主要提供用户查看某数据未来趋势功能。用户选择数据集并点击“预测”后，系统调用相应模型进行预测，并将结果通过ECharts折线图展示。图中展示了“A服务器内存使用量”未来30个时间点趋势走向，用户可以点击图形右上角按钮查看柱状图形和下载图形。

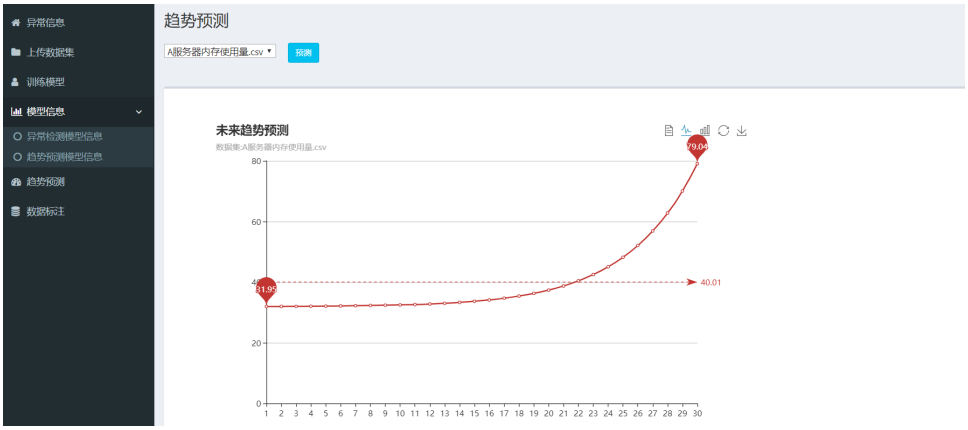

图 4.10: 趋势预测界面

异常信息界面用于集中显示异常检测模块检测到的异常实时数据，方便用户集中查看异常。数据标注界面用于提供批量数据标注功能，用户可以按数据集、时间区间和标签多条件查询数据，并通过单选、多选和全选的方式选中数据，然后点击“标签置为0”或“标签置为1”按钮批量更新数据标签。用户也可以选中数据集、时间区间和标签后点击“全部更改”按钮，系统会更新该数据集在这段时间区间内所有数据的标签，数据标注界面如图4.11所示。

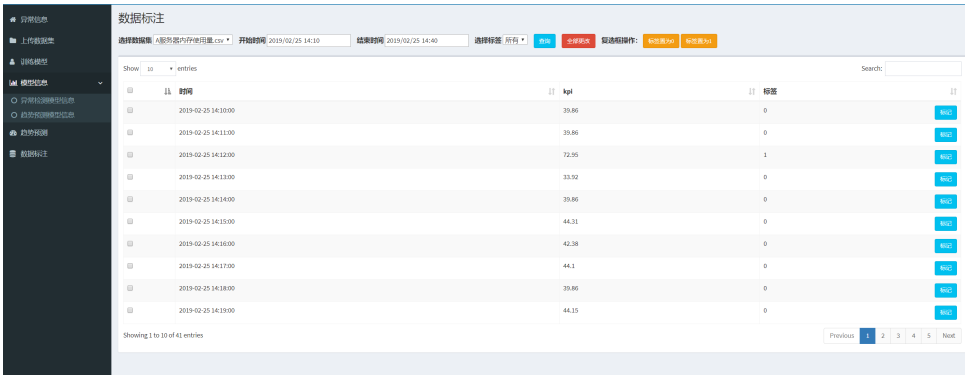

图 4.11: 数据标注界面

被监控系统产生实时数据后，异常检测模块接收数据并调用相应的异常检测模型对象进行异常检测，如发现异常会通过微信企业号和邮件报警。微信企

业号报警图如图4.12所示，报警分为阈值报警和异常检测模块报警两种，阈值报警消息显示“报警主机”、“主机IP”、“报警时间”等信息，异常检测模块报警消息显示“报警来源”、“报警时间”、“报警等级”和“报警信息”。用户通过查看微信可以及时掌握异常，在图4.12中，报警来源是“A 服务器内存使用量”，报警信息是实时数据信息，报警原因是A 服务器内存使用量达到89.16%。

图 4.12: 微信企业号报警

4.5.5　系统效果评估

本小节主要评估本系统异常检测和趋势预测的效果，通过对比传统固定阈值报警准确率和本系统异常检测准确率，评估本系统异常检测效果。通过对比数据真实趋势走向和本系统趋势预测数据，评估本系统趋势预测效果。

评估系统异常检测的准确率和漏报率。系统成功部署并正常运行30天，收集期间采用传统固定阈值报警的报警数、误报数和漏报数，收集本系统异常检测报警数、误报数和漏报数。通过对比两种报警方式的准确率和漏报率得出系统异常检测效果，30天内系统报警情况如表4.21所示。

表 4.21: 异常检测效果对比表

报警来源	报警总次数	异常次数	正常次数	漏报次数	准确率
固定阈值报警	83	49	34	23	59.04%
智能监控系统报警	67	58	9	14	86.57%

从表4.21中可以看出，采用固定阈值报警的方式在30天内产生了83次报警，其中正确报警49次，错误报警34 次，漏报23次，报警准确率为59.04%。智能监控系统在30天内产生了67次报警，其中正确报警58 次，错误报警9次，漏报14次，报警准确率为86.57%。通过对比可以看出，智能监控系统降低了9次漏报，减少了39.13%的漏报，提升了27.53%的准确率。

评估系统趋势预测的准确率。系统正常运行时，随机挑选10个数据集进行预测，记录每次预测的数值和真实数据的数值，通过对比两组数据的均方根误差得出趋势预测的准确性。实验时，随机挑选的数据集分别是A服务器的CPU使用率、磁盘使用率、网络流量流出，B服务器的内存使用率、网络流量流入，C服务器CPU使用率、进程数，MySQL连接数，网站在线人数和消息队列数。表4.22是10个数据集的趋势预测值与真实值之间的均方根误差值，由表可以得出，10个数据集的预测值与真实值之间的均方根误差的平均值为12.35，该均方根误差值较小，综上所述，本系统可以准确预测数据未来趋势走向。

表 4.22: 趋势预测效果表

数据集名称	均方根误差	数据集名称	均方根误差
A服务器内存使用率	3.74	A 服务器磁盘使用率	4.23
A服务器网络流出	21.35	B 服务器内存使用率	3.06
B服务器网络流入	19.32	C 服务器CPU使用率	2.61
C服务器进程数	12.02	MySQL连接数	20.74
网站在线人数	22.14	消息队列数	14.31

对比上述实验结果可以证明，本系统可以提供准确的异常检测与趋势预测功能，能够提高异常检测准确率，减少误报和漏报。

4.6　本章小结

本章详细介绍了基于XGBoost和LSTM的智能监控系统的具体实现。首先详细介绍了监控模块、数据处理模块、异常检测模块和趋势预测模块的主要功能、顺序图和关键代码实现，其中重点介绍了XGBoost算法参数确定实验、训练模型、重置模型和趋势预测模块的预测流程。然后设计测试用例对系统进行了功能测试和性能测试，通过多种测试验证系统的功能和性能，并展示了系统运行的效果图。最后通过对比实验评估了异常检测和趋势预测效果。

第五章　总结与展望

5.1　总结

为解决当前常用监控产品异常检测不准确和缺乏趋势预测功能的弊端，本文基于XGBoost和LSTM设计并实现了一套智能监控系统。该系统可以准确对实时数据进行异常检测，消除运维人员设置监控项阈值的工作量，提高系统异常检测准确率，减少误报警和漏报警。同时，该系统可以准确预测数据未来趋势走向，用户可以提前掌握数据趋势变化及早调度资源以保证系统稳定运行。本文主要工作如下：

（1）本文设计并实现了实时数据异常检测功能，实现了异常检测模型的训练、使用和重置功能。用户可以根据数据集训练多个互不相关的异常检测模型，系统可以根据实时数据来源调用不同的异常检测模型进行异常检测，如果发现异常数据则通过微信企业号和邮件通知相关运维人员。

（2）本文设计并实现了数据趋势预测功能，实现了趋势预测模型的训练、使用和重置功能。用户可以根据不同数据集训练多种不同的趋势预测模型，可以对数据进行趋势预测并通过折线图查看数据未来趋势走向。

（3）本文设计并实现了数据标注预处理功能，本文创新性地使用孤立森林算法对采集的监控数据进行数据标注预处理。用户上传数据集时，系统解析数据集并通过孤立森林算法对每条数据进行数据标注，以便于被监控模块识别和使用。数据标注预处理极大的减轻了运维人员数据标注的工作量，运维人员可以迭代更新数据标签并重置模型，不断提高异常检测模型的准确率。

（4）本文采用了Redis缓存和进程池提高系统性能，系统通过Redis缓存数据集名称和模型对象，提高模型对象读写速度，通过进程池后台异步训练和重置模型，保证系统响应速度。经过性能测试，本系统在面对高并发请求时，吞吐量保持稳定，能保证请求处理的及时性和正确性。

（5）本文评估了系统异常检测和趋势预测效果。经过30天的异常检测与趋势预测实验，结果证明本系统异常报警准确率达到86.57%，数据趋势预测均方根误差的平均值为12.35。与传统固定阈值报警方式相比，系统提高了27.53%的报警准确率，减少了39.13%的误报。由此可见，本系统可以提高异常检测准确率，减少误报和漏报，可以提供准确的异常检测和趋势预测功能。

5.2 展望

本系统已经成功部署并稳定提供异常检测与趋势预测服务，但还需要在以下方面进一步改进和完善。

（1）模型参数设置：在异常检测模块中，为了降低用户训练模型门槛，避免直接操作参数繁杂的算法，系统根据真实场景数据通过实验得出算法模型最优参数，并且对于不同数据集，使用这些参数训练的模型准确率较高。但是这些参数不一定完全适用于所有数据集，如果能根据数据集特征引导用户对模型参数进行设置，可能会提高每个异常检测模型的准确率。

（2）多模型融合：本系统异常检测和趋势预测算法模型单一，虽然准确率较高，但是如果同时使用多种算法配合的方式进行异常检测和趋势预测，可能异常检测与趋势预测准确率更高。

（3）运维生态完善：目前系统提供异常检测和趋势预测功能，希望未来能与智能根因分析、智能决策扩缩容、异常自动恢复等系统配合，形成更加智能和健全的运维体系，提供更加完善、智能和高效的功能。

参考文献

[1] W. Barth, Nagios: System and Network Monitoring, No Starch Press, 2008.

[2] D. Josephsen, Building A Monitoring Infrastructure with Nagios, Prentice Hall, 2007.

[3] M. L. Massie, B. N. Chun, D. E. Culler, The Ganglia Distributed Monitoring System: Design, Implementation, and Experience, Parallel Computing 30 (5-6) (2004) 817–840.

[4] P. Tader, Server Monitoring with Zabbix, Linux Journal 2010 (195) (2010) 7.

[5] 吴兆松, Zabbix 企业级分布式监控系统, 电子工业出版社, 2014.

[6] A. Dalle Vacche, Mastering Zabbix, Packt Publishing Ltd, 2015.

[7] L. Ghionna, G. Greco, A. Guzzo, L. Pontieri, Outlier Detection Techniques for Process Mining Applications, in: Proceedings of the Italian Symposium on Advanced Database Systems, 2008, pp. 263–270.

[8] E. M. Knorr, R. T. Ng, V. Tucakov, Distance-Based Outliers: Algorithms and Applications, The International Journal on Very Large Data Bases 8 (3-4) (2000) 237–253.

[9] F. Simmross-Wattenberg, J. I. Asensio-Pérez, P. Casaseca-de-la-Higuera, M. Martín-Fernández, I. A. Dimitriadis, C. Alberola-López, Anomaly Detection in Network Traffic Based on Statistical Inference and Alpha-Stable Modeling, IEEE Transactions on Dependable and Secure Computing 8 (4) (2011) 494–509.

[10] S. A. Dudani, The Distance-Weighted K-Nearest-Neighbor Rule, IEEE Transactions on Systems, Man, and Cybernetics 6 (4) (1976) 325–327.

[11] M. Köstinger, M. Hirzer, P. Wohlhart, P. M. Roth, H. Bischof, Large Scale Metric Learning from Equivalence Constraints, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2288–2295.

[12] D. Pokrajac, A. Lazarevic, L. J. Latecki, Incremental Local Outlier Detection for Data Streams, in: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, 2007, pp. 504–515.

[13] J. Tang, Z. Chen, A. W. Fu, D. W. Cheung, Enhancing Effectiveness of Outlier Detections for Low Density Patterns, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2002, pp. 535–548.

[14] M. M. Breunig, H. Kriegel, R. T. Ng, J. Sander, LOF: Identifying Density-Based Local Outliers, in: Proceedings of the ACM International Conference on Management of Data, 2000, pp. 93–104.

[15] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, C. Faloutsos, LOCI: Fast Outlier Detection Using the Local Correlation Integral, in: Proceedings of the International Conference on Data Engineering, 2003, pp. 315–326.

[16] D. Barbará, Y. Li, J. Lin, S. Jajodia, J. Couto, Bootstrapping A Data Mining Intrusion Detection System, in: Proceedings of the ACM Symposium on Applied Computing, 2003, pp. 421–425.

[17] M. Ester, H. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[18] A. K. Jain, Data Clustering: 50 Years Beyond K-Means, in: Proceedings of the Machine Learning and Knowledge Discovery in Databases.

[19] Z. He, X. Xu, S. Deng, Discovering Cluster-Based Local Outliers, Pattern Recognition Letters 24 (9-10) (2003) 1641–1650.

[20] 彭冬, 朱伟, 刘俊, 智能运维从0搭建大规模分布式AIOps系统, 电子工业出版社, 2018.

[21] 贾俊平, 何晓群, 金勇进, 统计学(第六版), 中国人民大学出版社, 2015.

[22] E. W. Saad, D. V. Prokhorov, D. C. W. II, Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks, IEEE Transactions on Neural Networks 9 (6) (1998) 1456–1470.

[23] J. Contreras, R. Espinola, F. J. Nogales, A. J. Conejo, ARIMA Models to Predict Next-Day Electricity Prices, IEEE Transactions on Power Systems 18 (3) (2003) 1014–1020.

[24] 郑为中, 史其信, 基于贝叶斯组合模型的短期交通量预测研究, 中国公路学报18 (1) (2005) 85–89.

[25] J. T. Connor, R. D. Martin, L. E. Atlas, Recurrent Neural Networks and Robust Time Series Prediction, IEEE Transactions on Neural Networks 5 (2) (1994) 240–254.

[26] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (8) (1997) 1735–1780.

[27] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, W. Woo, Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, in: Proceedings of the Advances in Neural Information Processing Systems, 2015, pp. 802–810.

[28] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese, Social LSTM: Human Trajectory Prediction in Crowded Spaces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 961–971.

[29] O. B. Sezer, A. M. Ozbayoglu, Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach, Applied Soft Computing 70 (2018) 525–538.

[30] W. Wang, Y. Shi, R. Luo, Sparse Representation Based Approach to Prediction for Economic Time Series, IEEE Access 7 (2019) 20614–20618.

[31] 郭晓慧, 李润知, 张茜, 王宗敏, 基于Zabbix 的分布式服务器监控应用研究, 通信学报34 (Z2) (2013) 94–98.

[32] A. Holovaty, J. Kaplan-Moss, The Definitive Guide to Django: Web Development Done Right, Apress, 2009.

[33] J. Forcier, P. Bissex, W. J. Chun, Python Web Development with Django, Addison-Wesley Professional, 2008.

[34] J. L. Carlson, Redis in Action, Manning Publications Company, 2013.

[35] F. T. Liu, K. M. Ting, Z. Zhou, Isolation Forest, in: Proceedings of the IEEE International Conference on Data Mining, 2008, pp. 413–422.

[36] T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System, in: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[37] S. Liu, J. Xiao, J. Liu, X. Wang, J. Wu, J. Zhu, Visual Diagnosis of Tree Boosting Methods, IEEE Transactions on Visualization and Computer Graphics 24 (1) (2018) 163–173.

[38] R. Mitchell, E. Frank, Accelerating the XGBoost Algorithm Using GPU Computing, PeerJ Computer Science 3 (2017) e127.

[39] N. Fitriah, S. K. Wijaya, M. I. Fanany, C. Badri, M. Rezal, EEG Channels Reduction Using PCA to Increase XGBoost' s Accuracy for Stroke Detection, in: Proceedings of the American Institute of Physics Conference Series, Vol. 1862, AIP Publishing, 2017, p. 030128.

[40] X. Ren, H. Guo, S. Li, S. Wang, J. Li, A Novel Image Classification Method with CNN-XGBoost Model, in: Proceedings of the International Workshop on Digital Watermarking, 2017, pp. 378–390.

[41] D. Zhang, L. Qian, B. Mao, C. Huang, B. Huang, Y. Si, A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGBoost, IEEE Access 6 (2018) 21020–21031.

[42] M. Ture, F. Tokatli, I. Kurt, Using Kaplan-Meier Analysis Together with Decision Tree Methods (C&RT, CHAID, QUEST, C4.5 and ID3) in Determining Recurrence-Free Survival of Breast Cancer Patients, Expert Systems with Applications 36 (2) (2009) 2017–2026.

[43] A. Pulver, S. Lyu, LSTM with Working Memory, in: Proceedings of the International Joint Conference on Neural Networks, 2017, pp. 845–851.

简历与科研成果

基本情况 张双江，男，汉族，1994 年 10 月出生，湖北省宜昌市人。

教育背景

2017.9～2019.6 南京大学软件学院 硕士

2013.9～2017.6 华中科技大学软件学院 本科

参与项目

1. 中华人民共和国南京海关：运行数据可视化研究相关技术支持服务，2018-2019
2. 南京南瑞集团项目：移动众测平台软件（20170413），2017-2018

致　　谢

惊风飘白日，光景驰西流。昨日仿佛才开始研究生生活，今日便要别离母校各奔东西，借此机会我要感谢所有帮助支持关心我的人，感谢你们的关爱。

首先要感谢我的导师陈振宇老师，研究生期间，陈老师在项目上、技术上、生活上给我莫大的帮助，全程对我的毕设和论文给予指导和帮助。感谢项目初期陈老师帮我整理思路、指明可行性方向，多次通过会议跟踪我们项目进展并提出宝贵意见，不断改进项目和论文，督促我们及时完成项目和论文。陈老师身体力行，精力充沛的投入产学研的实干精神鼓舞着我要务实和好学。感谢房春荣老师在项目实践阶段提出了许多宝贵意见。

感谢我的实验室“iSE实验室”，感谢实验室提供各种丰富的资源和平台供我学习使用和成长。感谢实验室同学的勤奋学习和刻苦科研为我提供融洽好学的学习氛围，感谢你们在会议上分享的各种技术研究让我大开眼界。

感谢实验室的黄勇老师在技术上对我的帮助和指导。感谢梅杰、李秋婷、赵文远、陈笑智、李沛源和周顺祥同学，做毕设的时候我们能够集思广益、相互帮助、互相激励监督，沟通解决了很多项目中的问题，感谢你们的陪伴。

我要感谢南京大学，两年的研究生生活充实又短暂，感谢南京大学提供的平台，校训“励学敦行，诚朴雄伟”将永远留在我心中，激励我砥砺前行。

我要感谢我的父母和家人对我的支持和帮助，家庭是我最坚实的后盾，父母给我的力量会永远激励我向着自己的理想不断前行。感谢研究生阶段所有帮助过我的老师、同学、同事和朋友。

最后，我要感谢各位论文审稿的老师和专家，向你们的专业和勤劳致敬!

版权及论文原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：

日期：　　年　　月　　日