南京大學

研 究 生 毕 业 论 文

（申请硕士专业学位）

论 文 题 目	基于模糊测试的推荐模型质量分析系统的设计与实现
作 者 姓 名	曹可凡
专 业 名 称	软件工程
研 究 方 向	软件工程
指 导 教 师	陈振宇 教授，房春荣 助理研究员

2021 年 5 月 20 日

学　　号：**MF1932007**

论文答辩日期：**2021 年 5 月 20 日**

指 导 教 师：　　**（签字）**

The Design and Implementation of Recommendation Model Quality Analysis System Based on Fuzz Testing

By

Kefan Cao

Supervised by

Professor **Zhenyu Chen**

Assistant Professor **Chunrong Fang**

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 2021

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：基于模糊测试的推荐模型质量分析系统的设计与实现

软件工程 专业 2019 级硕士生姓名：曹可凡

指导教师（姓名、职称）：陈振宇 教授，房春荣 助理研究员

摘 要

在信息爆炸的时代，推荐系统对于互联网公司至关重要，例如 YouTube 利用推荐系统实现从上亿视频中针对不同用户的兴趣点生成相应的推荐视频列表。因此，保障推荐系统的模型质量成为非常有价值的研究问题。现有的推荐模型测试方法采用量化评估指标分析用户历史数据，一方面历史数据诱导推荐结果过分倾向于用户既有选择，使得推荐多样性降低，造成历史数据偏见。另一方面，测试采用的评估指标不够丰富，造成推荐模型质量评估不全面。这些问题导致现有测试方法无法满足开发者进行推荐模型质量评估的需要。因此需要开发高质量的推荐模型质量分析系统，有效发掘推荐系统内部模型的质量缺陷。

为此，本文设计并实现了基于模糊测试的推荐模型质量分析系统，通过模糊测试方法扩增测试数据，并将量化指标与可解释性分析相结合实现模型质量评估。本系统考虑到推荐场景下特征形式丰富，且特征间存在关联性，采用特征结构创建模糊测试语料集，记录特征依赖关系，并规定特征类型和相应变异方法。本系统构建的语料集基于特征类型转化并存储原始测试数据，同时完成数据规模统计与特征分布统计。本系统提供了丰富的内置变异方法，基于语料集记录的特征依赖关系，能够针对文本、数值、字典、列表等不同形式的测试用例特征值进行调整，从而生成新测试用例并添加到语料集中，实现测试数据扩增。扩增数据集将作为推荐系统的测试输入。本系统为了解决现有评估指标单一化问题，提供了热度值等特色评估指标，能够有效衡量推荐丰富性，并采用 t-SNE 降维等可解释性指标分析特征工程质量。本系统由数据流模块、模糊测试任务模块、任务管理模块和报告生成模块组成。数据流模块向使用者提供测试文件上传和扩增数据下载功能，模糊测试任务模块负责完成特征结构创建、数据扩增和指标计算，任务管理模块为使用者提供任务信息查看功能，报告生成模块负责利用指标数据和测试任务信息完成自动化测试报告生成。

基于模糊测试的推荐模型质量分析系统已在蚂蚁集团试运行，并针对 8 个业务场景下的具体模型进行质量评估。本系统通过量化指标发掘了 8 个模型存在质量缺陷，通过可解释性指标发掘 12 个模型存在特征工程缺陷，实用性较高。

关键词：推荐模型，模糊测试，评估指标，数据扩增

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 Recommendation Model Quality Analysis System Based on Fuzz Testing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Kefan Cao

MENTOR: Professor **Zhenyu Chen**
Assistant Professor **Chunrong Fang**

Abstract

In the era of information explosion, recommendation systems are crucial for Internet companies. For example, YouTube utilizes it to provide videos from hundreds of millions of resources for different users based on their personal interests. Therefore, guaranteeing the quality of recommendation models becomes a very valuable research problem. Existing recommendation model testing methods use quantitative evaluation metrics to analyze users' historical behavioral data. On the one hand, historical data induces recommendation results to be overly inclined to existing choices, which reduces recommendation diversity and causes historical data bias problem. On the other hand, the evaluation metrics are not rich enough, resulting in incomplete quality assessment of the recommendation model. These problems make the existing testing methods unable to meet the needs of developers for recommendation model quality evaluation. Therefore, there is a need to develop a high-quality recommendation model quality analysis system to discover the deep quality defects of the recommendation model and improve the testing adequacy.

Therefore, this thesis designs and implements a recommendation model quality analysis system based on fuzz testing, which augments test cases by fuzz testing technology and combines quantitative metrics with interpretable analysis to complete recommendation quality evaluation. Considering diverse forms of features in recommendation scenarios and the relationship between features, the system utilizes feature structure to create a corpus to record feature correlations, and to specify feature types and mutation methods. The corpus constructed by this system transforms and stores the

original testing dataset based on feature types, and completes data scale statistics and feature distribution analysis. Based on the feature correlations recorded in the corpus, the system provides rich built-in mutation methods to adjust the feature values of test cases in different forms, such as text, numeric, dictionary, and list, to generate new test cases and add them to the corpus to augment testing dataset which will be used to evaluate the recommendation system. In order to rich evaluation metrics, the system provides characteristic evaluation metrics such as heat value, which can effectively measure recommendation diversity, and analyzes the quality of feature engineering based on t-SNE dimensionality reduction. The system consists of a data flow module, a fuzz testing task module, a task management module and a report generation module. The data flow module supports test file uploading and data downloading; the fuzz testing task module is responsible for feature structure creation, data augmentation and metrics calculation; the task management module provides users with task information; the report generation module is responsible for automated test report generation.

The system has been piloted within Ant Group and has been used to evaluate the quality of models under eight domains. This system has uncovered 8 models with quality defects through quantitative metrics and 12 models with feature engineering defects through interpretability analysis metric, which shows high practicality of the system.

Keywords: Recommendation model, fuzz testing, evaluation metrics, data augmentation

目录

表 目 录 ····· **x**

图 目 录 ····· **xii**

第一章 引言 ····· **1**

1.1 课题背景和意义 ····· 1

1.2 相关研究现状 ····· 2

1.2.1 推荐模型核心技术研究现状 ····· 2

1.2.2 推荐模型测试技术研究现状 ····· 3

1.2.3 基于对抗攻击的推荐模型测试技术研究现状 ····· 4

1.3 本文主要工作 ····· 5

1.4 本文组织结构 ····· 6

第二章 技术综述 ····· **7**

2.1 模糊测试 ····· 7

2.2 常见推荐算法 ····· 9

2.2.1 协同过滤算法 ····· 9

2.2.2 MF、LR 和 FM 算法 ····· 10

2.2.3 LS-PLM ····· 11

2.2.4 Wide&Deep ····· 11

2.3 前端技术 ····· 11

2.3.1 Vue ····· 12

2.3.2 ElementUI ····· 13

2.4 后端技术 ····· 13

2.4.1 Django ····· 13

2.4.2 scikit-learn ····· 15

2.5 本章小结 ····· 15

第三章 系统分析与设计 ····· **17**
3.1 系统整体概述 ····· 17
3.2 系统需求分析 ····· 17
3.2.1 涉众分析 ····· 17
3.2.2 系统功能性分析 ····· 18
3.2.3 系统非功能性分析 ····· 19
3.2.4 系统用例图 ····· 20
3.2.5 系统流程图 ····· 26
3.3 系统总体设计 ····· 27
3.3.1 模块说明 ····· 27
3.3.2 4+1 视图 ····· 28
3.4 持久化设计 ····· 32
3.5 本章小结 ····· 36

第四章 变异方法与评估指标 ····· **37**
4.1 特征分类设计 ····· 37
4.1.1 数值型 ····· 37
4.1.2 字符型 ····· 38
4.1.3 布尔型 ····· 40
4.1.4 列表型 ····· 40
4.1.5 字典型 ····· 41
4.2 变异方法设计 ····· 41
4.2.1 数值型 ····· 41
4.2.2 字符型 ····· 42
4.2.3 列表型 ····· 43
4.2.4 字典型 ····· 43
4.2.5 类型无关方法 ····· 44
4.3 评估指标设计 ····· 44
4.3.1 准确率 ····· 44
4.3.2 精准率和召回率 ····· 44
4.3.3 平均精度均值 ····· 45

4.3.4 热度值与热度分桶 ····· 45
4.3.5 t-SNE 降维 ····· 46
4.4 本章小结 ····· 46

第五章 系统详细设计与实现 ····· 47
5.1 数据流模块设计与实现 ····· 47
5.1.1 流程设计 ····· 47
5.1.2 时序图 ····· 49
5.1.3 关键代码 ····· 50
5.1.4 界面演示 ····· 52
5.2 模糊测试任务模块设计与实现 ····· 53
5.2.1 流程图 ····· 53
5.2.2 时序图 ····· 55
5.2.3 关键代码 ····· 57
5.2.4 界面演示 ····· 58
5.3 任务管理模块设计与实现 ····· 60
5.3.1 流程图 ····· 60
5.3.2 时序图 ····· 60
5.3.3 界面演示 ····· 61
5.4 报告生成模块设计与实现 ····· 61
5.4.1 流程图 ····· 61
5.4.2 时序图 ····· 62
5.4.3 关键代码 ····· 63
5.4.4 界面演示 ····· 63
5.5 本章小结 ····· 64

第六章 系统测试与分析 ····· 65
6.1 测试环境 ····· 65
6.2 单元测试 ····· 66
6.3 性能测试 ····· 67
6.4 案例演示 ····· 68
6.4.1 文件上传 ····· 68

6.4.2 特征结构创建 ····· 69
6.4.3 测试任务创建 ····· 70
6.4.4 测试任务查看 ····· 71
6.4.5 评估报告查看 ····· 72
6.4.6 测试数据下载 ····· 73
6.5 系统有效性验证 ····· 73
6.5.1 实验设置 ····· 74
6.5.2 对比分析 ····· 75
6.5.3 试运行结果分析 ····· 75
6.6 本章小结 ····· 76

第七章 总结与展望 ····· **77**
7.1 总结 ····· 77
7.2 展望 ····· 78

参考文献 ····· **79**

简历与科研成果 ····· **83**

致谢 ····· **85**

表 目 录

2.1 MTV 模式 …… 14

3.1 系统涉众分析 …… 18
3.2 系统功能需求列表 …… 18
3.3 用户注册与登录用例描述 …… 21
3.4 创建特征结构用例描述 …… 22
3.5 创建测试任务用例描述 …… 23
3.6 上传数据用例描述 …… 24
3.7 查看任务信息用例描述 …… 25
3.8 获取测试报告用例描述 …… 25
3.9 下载扩增数据用例描述 …… 26
3.10 用户信息表 …… 34
3.11 模型文件表 …… 34
3.12 测试任务表 …… 34
3.13 特征结构表 …… 35
3.14 数据集表 …… 35
3.15 评估结果表 …… 35
3.16 评估散点表 …… 36
3.17 扩增数据下载表 …… 36

4.1 数值型统计器 …… 38
4.2 字符型统计器 …… 39
4.3 布尔型统计器 …… 40
4.4 列表型统计器 …… 40
4.5 字典型统计器 …… 41

6.1 测试环境信息 …… 65
6.2 测试环境信息 …… 66

6.3 性能测试结果 ····· 68
6.4 DELF 模型测试结果 ····· 75
6.5 系统试运行结果 ····· 76

图 目 录

2.1 模糊测试基本流程 …… 8
2.2 Vue 结构示意图 …… 12

3.1 系统用例图 …… 20
3.2 系统流程图 …… 26
3.3 系统架构图 …… 27
3.4 逻辑视图 …… 29
3.5 开发视图 …… 30
3.6 进程视图 …… 31
3.7 物理视图 …… 32
3.8 系统 E-R 图 …… 33

5.1 文件上传流程图 …… 47
5.2 文件下载流程图 …… 48
5.3 文件上传时序图 …… 49
5.4 文件下载时序图 …… 50
5.5 上传流程关键代码 …… 51
5.6 下载流程关键代码 …… 52
5.7 文件上传页面展示 …… 52
5.8 文件下载页面展示 …… 53
5.9 特征构建流程图 …… 54
5.10 任务创建流程图 …… 55
5.11 特征构建时序图 …… 56
5.12 任务创建时序图 …… 57
5.13 构建特征结构关键代码 …… 58
5.14 特征创建页面展示 …… 59
5.15 任务创建页面展示 …… 59
5.16 任务管理流程图 …… 60

5.17 任务管理时序图 ····· 61
5.18 任务管理页面展示 ····· 62
5.19 报告生成流程图 ····· 62
5.20 报告生成时序图 ····· 63
5.21 报告生成关键代码 ····· 64
5.22 报告生成页面展示 ····· 64

6.1 系统单元测试结果 ····· 67
6.2 上传数据集、数据集读取文件和模型 ····· 68
6.3 构建特征结构 ····· 69
6.4 新建测试任务 ····· 70
6.5 测试任务查看 ····· 71
6.6 评估报告查看 ····· 72
6.7 测试数据下载 ····· 73

第一章　引言

1.1　课题背景和意义

在数据泛滥的时代，用户正面临着一种新的信息爆炸形式。为了在缓解用户决策焦虑，加强用户体验的同时，为服务提供方提高平台流量，延长页面停留时间或增加商品交易量，推荐系统（RS）应需而生。在具体应用场景下，推荐系统针对用户“喜好”进行个性化定制，提供令用户满意的推荐服务。可以说，当今的推荐系统，已经成为人们日常生活中难以分割的，大多数软件应用背后不可或缺的核心技术。因此推荐系统也是各大互联网企业（如 Facebook [1]，Google [2]，YouTube [3]，Netflix[4]，AliBaba[5]，Tencent[6] 等）的建设重点。

推荐系统这一概念是由 Karlgren 于 1990 年在其研究报告中 [7] 首次提出。早期的推荐系统采用传统机器学习模型作为核心算法，且其技术演变大致经历了 4 个关键阶段，即协同过滤（CF）算法阶段、逻辑回归（LR）阶段、因子分解机（FM）阶段和组合模型阶段。在 2016 年后，YoutubeDNN[8] 推动了推荐系统的技术革新，在计算机视觉（CV）和自然语言处理（NLP）领域已有丰富积累的深度学习技术开始冲击当时的推荐系统体系，采用深度学习模型替代和融合既有传统机器学习模型，不仅是学界的全新热门研究方向，也是被各大企业在实际业务中应用的解决方案。与传统机器学习模型相比，深度学习模型具有表达能力更强，模型结构更灵活，业务场景更贴合等诸多优势 [9]。

尽管如此，推荐系统并不能在所有推荐场景下令用户满意。除了推荐服务本身的性能问题和信息文案来源的丰富性外，推荐系统背后所依赖的算法模型也可能存在缺陷，因此需要对推荐模型进行全方位的测试。目前针对推荐模型的测试方法主要包括离线测试方法、内部评测/仿真测试方法和在线测试方法。离线测试方法主要依托用户历史行为（如用户已购买商品）生成数据集，并采用多种离线评估指标分析新模型的质量。内部评测一般采取邀请或雇佣用户的方式对推荐模型质量进行人为评估，收集问卷反馈，而仿真测试则将数据集基于时间进行分割，实现对用户未来行为的模拟。在线测试方法直接依靠生产环境，测试过程采用的典型策略是 AB 测试，通过收集和分析用户在采用了旧模型和新模型的两套推荐系统下的行为差异，评估推荐系统是否可采用新模型。

目前针对推荐模型的测试方法存在一定缺陷。离线测试方法除了评估指标单一问题外，还会忽视推荐模型的历史数据偏见问题。历史数据偏见是指推荐

结果与用户历史选择过于相近，造成推荐丰富性差，且无法有效应对用户未来的需求变化。例如某用户仅有多次计算机类相关书籍购买记录，但未来可能会购买笔记本电脑。由于历史数据不包含这一信息，导致离线测试方法无法揭露该问题，从而造成系统缺陷发掘有限。内部评测方法调查成本较高，用户规模有限，且最终测评结果存在很强的主观性。在线测试方法周期较长，需要进行较长时间的用户体验反馈收集分析才能完成质量评估过程。因此，推荐模型测试存在降低测试成本，提高测试充分性，减少开发者工作成本的需求。

为此，本文设计并实现了基于模糊测试的推荐模型质量分析系统。系统收集并存储开发者提供的测试数据集和模型文件，并采用特征结构机制管理相关特征信息，通过可视化组件降低了开发者建立特征关联关系的操作复杂性，实现了多测试任务下的特征关系复用。系统采用模糊测试技术构建测试流程，利用开发者创建的特征结构和提供的原始数据集，通过丰富的内置特征类型和变异方法，完成测试数据扩增和执行，提高了测试充分性。系统为开发者提供了自动化测试报告生成服务，通过结合特色评估指标和可解释性分析过程，向开发者反馈推荐模型质量，更好地帮助开发者发现模型缺陷，推进优化工作。

1.2　相关研究现状

1.2.1　推荐模型核心技术研究现状

推荐模型从协同过滤到组合模型，可解释性不断降低。所谓可解释性，是指通过某种方式理解模型的决策机理，并能够以此调整模型相应参数来弥补模型缺陷。随着 2016 年后深度学习技术的广泛应用，传统机器学习方法在企业界逐渐被融合和替代，而模型解释难度也进一步上升。

YoutubeDNN[3] 代表了现今企业采用的推荐系统架构典型模式。召回和排序是推荐系统内部模型的两个基本工作模块。其中召回模块负责快速筛选出一批用户最感兴趣的待选对象，其目标是保证高召回率（也称查全率）。而排序模块则将召回结果按照用户的兴趣期望排序，其目标是在 TopN（即前 N 个推荐物品）下高精准率或高用户满意度。无论是召回模块还是排序模块，都包含特征 ID 化这一步重要预处理工作。特征 ID 化的主要目的是解决原始特征映射处理后特征向量（如 one-hot 表示）稀疏问题，采用全连接（dense）方式将稀疏特征转化为稠密特征。召回模块考虑到候选集庞大（百万级到千万级），为了保证处理效率选用较少的用户特征，同时并非所有特征都进行 ID 化处理，如出现性别等取值有限的特征，或是地理位置等已经稠密化处理的特征，可以不进行全连接处理，而后这批被称为 Embedding 的处理结果将被拼接（concat) 为一个特征向

量，并通过三层 ReLU 处理，最后通过 softmax 计算用户类别，并采用最邻近算法获得推荐物品候选集，并交给排序模块进行精排。在这一过程中，Embedding 作为 ID 化产物，是深度学习（deep learning）训练过程的重要学习参数。排序模块的工作原理与推荐模块类似，但由于精排需要更准确和更具可比性的打分或排名，因此构建的 Embedding 需要考虑更多的用户/物品特征。为了降低 ID 化流程时间，作者专门提出同类特征（如观看视频 ID 集和收藏视频 ID 集）共享 Embedding，并保证了在特征向量拼接过程中同类特征仍然享有独立空间。

Cheng H[10] 等人提出了 Wide & Deep 架构，将特征处理过程分成 wide 和 deep 部分。wide 部分采用线性模型，将原始特征和人工构建的叉乘特征进行特征组合，从而为线性模型引入非线性特性。deep 部分与 YoutubeDNN 类似，训练深度学习模型获取 Embedding 表示，基于 Embedding 表示构建稠密特征向量，最终基于稠密特征向量完成排序。Wide & Deep 架构对 wide 部分和 deep 部分进行联合训练，同时完成参数优化，从而将线性模型的记忆能力和深度学习模型的泛化能力相结合，提高了模型的整体性能。针对 Wide & Deep 结构有许多优化工作，本文将在第二章技术综述第二节进行详细介绍。

1.2.2　推荐模型测试技术研究现状

推荐模型的测试工作与推荐模型的改进优化同等重要，王在其著作 [9] 中提出了三个切实理由：（1）测试阶段所采用的评估指标是推荐模型的优化提升方向；（2）测试评估报告是不同团队间接口性工作的沟通基石；（3）测试指标的选取是企业的商业目标反映和价值观体现。除此之外，推荐模型的测试工作也是对系统稳定性和有效性的保证。

传统软件系统测试技术一直都是测试领域的热门话题，针对不同应用场景、不同编程语言、不同结构单元的测试工具和测试理论层出不穷 [11][12][13]。然而，相比于传统软件系统的测试评估工作，量化指标对推荐模型的测试评估过程更重要，主要因为推荐模型输入空间庞大，且反馈结果不可预期，因此需要利用多种指标从不同维度来评估推荐模型质量。

离线测试方法建立在从原始数据集中分割出的测试集上。由于离线环境与线上环境隔离，离线测试方法具有风险低成本小，测试速度快的优势。这一过程中的常用指标是在 TopN 下的精准率和召回率、ROC 曲线、准确率和覆盖率等。离线测试方法能够快速排查模型的质量缺陷。

仿真测试方法与离线测试方法一样在线下环境进行，但在数据集处理上存在差异。L. Li[14] 等人提出 Replay 方法（动态离线评估方法），将日志信息按照

生成日期进行分割，从而生成多个测试子集。每次训练迭代中，都将上一次测试集加入训练集重新训练，并用当前测试集进行指标评估，从而模拟用户实际行为，有效覆盖了更多测试场景，提高了测试充分性。

在线测试方法在实际生产环境下对推荐系统的模型质量进行评估。AB 测试是在线测试的典型模式 [15]，通过对用户进行分流，对比分析新旧系统下的用户反馈差异，从而评估新系统是否比旧系统更优秀。RadlinskiF 等人 [16] 为了解决 AB 测试过程中用户分组可能存在的不平衡问题，提出了 Interleaving 方法。该方法放弃了用户分组，而是在推荐结果列表中混合新旧系统的推荐结果，一并向用户展示，并根据用户偏好的统计结果对新旧系统的质量进行对比。尽管 Interleaving 比传统 AB 测试速度更快，用户体感差异小，但由于只能获取用户偏好这一表现，不如 AB 测试全面，因此一般在线上测试过程中将两者相结合。

1.2.3　基于对抗攻击的推荐模型测试技术研究现状

对抗攻击作为一种评估神经网络模型是否具有缺陷的测试方法，首先在计算机视觉领域（CV）被提出，并主要用于发掘可能由模型识别错误触发的潜在安全隐患 [17][18][19]。在计算机视觉领域，对抗攻击的基本思想是对输入的图片进行“细微”扰动，在保证其干扰程度肉眼不可见（或是语义不变）的前提下，使神经网络模型的预测或判断产生错误。对抗攻击的相关研究在图像领域逐渐成熟，许多研究者将目光转向推荐领域，希望能够利用现有研究成果分析或解释深度学习模型在推荐领域下的缺陷，帮助模型优化。

推荐领域并没有 CV 领域的安全性问题，然而这并不代表对抗攻击方法在推荐模型测试中无用武之地。由于模型训练所用的用户交互数据可能被用户误操作或恶意修改等行为，混入不可去除的噪声，从而导致模型的推荐质量评估指标下滑。同时，由于对抗攻击方法能够有提升模型鲁棒性和提供一定可解释性，因此对抗攻击方法也是优化推荐系统模型质量的途径之一。

除了上述推荐领域与计算机视觉领域上对抗攻击的共性，攻击方法在向推荐领域的迁移过程中仍然有很多障碍。推荐模型的原始输入通常为用户和物品特征信息，其中包含了图片、文本、数值、关系等多种数据形式，不仅特征数据结构更为复杂，计算机视觉领域中对“微小”扰动概念的定义也无法直接应用到推荐领域。为了解决这些问题，部分研究者从 Embedding 出发，利用 Embedding 自身的矩阵结构和数值连续的特点，将计算机视觉领域中基于梯度下降的攻击方法进行迁移。何等人提出 [20] APR 对抗攻击体系，针对排序过程中常用的基于矩阵分解的贝叶斯推荐模型 (MF-BPR)，设计了 min-max 对抗训练流程。在训练过程中首先最小化自定义对抗攻击损失函数，再最大化模型损失函数，训练过

程经过多轮迭代后得到收敛。唐等人 [21] 的攻击方法与何等人的类似，但攻击方法并未选择直接在 Embedding 或原始特征上进行修改，而是专门训练了一个深度神经网络对原始特征进行抽取转化，并在转化后的特征上进行扰动。这种攻击方式更接近原始输入，同时也提高了攻击训练效率。王等人 [22] 提出 IRGAN 将 GAN 体系成功引入到推荐领域，针对检索应用场景下的输入序列这一原始特征构建生成器进行攻击，同时利用判别器分析输入是否为生成的虚假数据，通过二者博弈完成模型优化。

尽管基于对抗攻击的推荐模型测试技术在模型质量评估工作中取得了一定成效，但由于测试过程需要调整推荐模型结构，且包含复杂耗时的推荐模型重训练工作，造成测试成本过高，无法满足开发者的实际需要。

1.3　本文主要工作

现有的推荐模型质量评估方法存在忽视历史数据偏见、评估指标单一、测试不充分等诸多缺陷。为了解决这些问题，本文设计并实现了基于模糊测试的推荐模型质量分析系统。开发者能够利用该系统完成测试数据集上传，快速构建模糊测试任务，查看任务执行状态和任务详情信息，以及获取测试质量评估报告。系统除了提供丰富的内置特征划分、变异方法库和评估指标，还支持开发者上传自定义特征类型文件和变异方法文件，满足个性化测试需求。

为了达成上述功能，满足开发者需要，本文完成了以下工作：

（1）借鉴以往深度学习系统模糊测试 [22][23][24] 的研究思路，提供了规模庞大的系统内置特征类型和变异方法。不同于以调整图片像素值和图片规模为主的变异方法，本系统的内置方法库适配.json、.dat、.csv 等常见数据文件格式，并支持数值型、字符型、布尔型、列表型和字典型的特征调整。由于目前推荐模型大多不选择直接将图片等复杂文件作为特征值进行训练，而是在其基础上抽取出高维特征向量进行替代，因此变异方法库暂不支持对多媒体数据的变异扩增。扩增数据用于揭露历史数据偏见等推荐模型质量问题。

（2）提供了丰富的内置评估指标。除了包含精准率、召回率、MAP 等常用的推荐模型质量评估指标，系统还提供了诸如热度值等额外评估指标。通过分析模糊测试过程的执行结果，系统完成指标运算，并将其渲染为测试报告向开发者展现。同时为了进行更深入的模型质量分析，系统还提供了基于用户和物品 Embedding 向量值的 t-SNE 降维散点图进行可解释性分析，通过展现 Embedding 的聚集性，帮助开发者优化推荐模型特征工程，提高推荐模型质量。

（3）支持特征结构创建。本系统支持开发者在可视化操作界面下，通过拖拽

和连线的方式，描述推荐场景中不同特征间的关联性，如用户的生日特征和星座特征。同时，开发者提供特征名称、特征类型、变异方法等相关参数。最终，特征结构将被系统转化为持久化结构记录。在模糊测试过程中，系统利用相应特征结构完成语料集创建，并采用特征结构规定的特征类型和变异方法调整测试用例特征值，完成测试数据的变异扩增。

1.4　本文组织结构

本文总共分为七个章节，各章节的主要内容如下：

第一章，引言。本章主要介绍了基于模糊测试的推荐模型质量分析系统的课题背景与研究意义，梳理了推荐模型、推荐模型测试和对抗攻击推荐模型测试技术的研究现状，阐明了在推荐模型测试领域存在的问题。最后介绍了本文为解决推荐模型测试领域的现有问题完成的主要工作。

第二章，技术综述。本章首先介绍了模糊测试技术的基本流程和技术优势。然后介绍了本系统支持的常见推荐算法与模型结构，最后介绍了本系统前后端实现所选取的关键技术以及技术优势。

第三章，系统分析与设计。本章首先给出系统的功能与非功能性需求，以及相应的用例图和用例描述，接着给出本系统的整体架构设计图与模块划分，并利用 4+1 视图剖析系统架构组织方式，最后基于 E-R 图所描述的系统实体关系，给出系统的持久化设计方案。

第四章，变异方法与评估指标。本章首先介绍了系统内置的特征分类，包括特征划分依据和适用范围，以及开发者构建特征所需提供的额外参数。接着对内置变异方法库包含的具体变异方法进行说明，包括方法适用场景、实现原理和相关参数。最后介绍了内置评估指标的实现原理。

第五章，系统详细设计与实现。本章针对数据流模块、模糊测试任务模块、任务管理模块和报告生成模块进行详细说明，分别介绍了其所对应的流程图和时序图，并通过关键代码补充实现细节，最后展示相应页面截图并进行说明。

第六章，系统测试与分析。本章首先给出了系统单元测试和性能测试结果，证明了系统各模块符合开发预期，且满足非功能性需求。然后设计并演示了一则详细案例，通过展示系统核心链路操作流程，阐述了系统使用方法，同时证明系统满足了功能性需求。最后本文给出了系统有效性验证，包括一组对比实验和系统试运行过程中发现的模型缺陷。

第七章，总结与展望。本章总结了系统的主要工作，描述了本系统目前达到的测试效果，并提出本系统当前存在的不足与未来的工作方向。

第二章　技术综述

2.1　模糊测试

模糊测试最早由 Miller[25] 等人提出，用于对 UNIX 程序的完备性通过命令行进行测试，其核心思想是通过自动或半自动方式生成大批测试用例，并利用异常捕获等检测方法分析用例的执行结果。模糊测试的主流模式是黑盒测试，即完全不关注软件的内部实现与组织架构，只关心输入的合法性与输出的合理性。也有部分测试实践会选择灰盒测试模式，并在测试过程中获取更多的中间数据作为系统的质量评估补充。模糊测试方法如今已经在传统测试领域获得了充分发展，且已经证明了其系统漏洞查找效率，AFL (American Fuzzy Lop. [1]) 便是最著名的模糊测试工具之一。

如图2.1所示，模糊测试过程主要包括如下几步。(1) 语料集构建。为了向待测系统输入自动化构建的测试数据，模糊测试过程需要设定测试语料集。语料集除了用于存储构建的测试用例，还负责检测测试用例是否符合输入规范，并对生成测试用例的规模和资源消耗进行统筹规划，从而保证用例生成的有效性；(2) 测试用例生成。常见的生成策略分为两种，即变异模式（Mutate）和生成模式（Generate）。变异模式需要原始测试数据和变异规则，并依照变异规则对原始数据的部分或全部进行修改，以此生成新的测试用例。而生成模式则需要构建用例生成规则，并根据测试人员自定义种子或用例生成参数，直接得到新测试用例。变异模式无需完整的用例构建规则，但需要采集已有测试用例；生成模式无需依赖现有测试用例，但人工成本较高，且用例生成丰富性可能不足。(3) 待测系统执行。完成系统环境部署后，从语料集中抽取测试用例并执行。执行过程中完成软件监控，异常信息通过输出透出或灰盒模式下进行中间过程分析。(4) 用例评估。用例评估并非必要步骤，但如果是引导模式下的模糊测试（如覆盖引导的模糊测试），一般会收集额外信息进行扩增用例评估，并在语料集中只保留“有价值的”（如覆盖了更深层次的软件执行分支）扩增数据，从而提升语料集的整体价值，提高其发现更深层软件缺陷的潜力。除此之外，触发了异常缺陷的用例的详细信息都会被记录和分析，但会被测试系统从语料集中移除。完成了该步骤之后，模糊测试系统从第（2）步开始循环执行，直到测试达到设定规模。

对于深度学习系统，尤其是计算机视觉领域下相关系统的测试工作，早期

[1] http://lcamtuf.coredump.cx/afl.

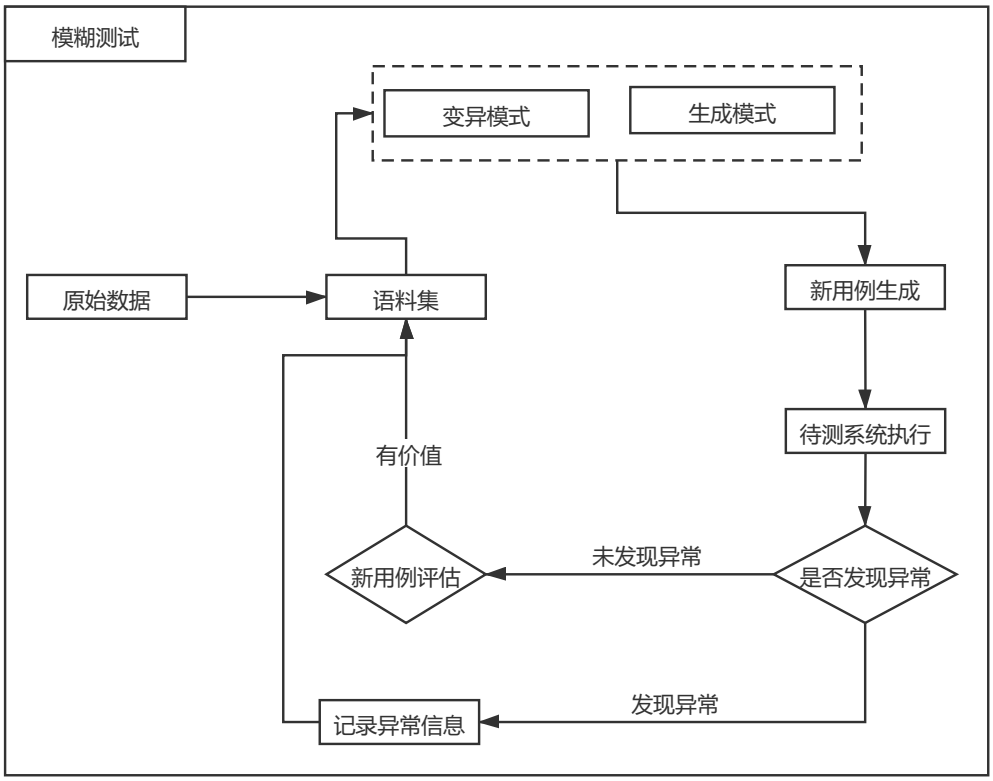

图 2.1: 模糊测试基本流程

被作为模型论文实验的一部分，并以模型判断准确率作为主要关注指标。显然这样的测试方法是不充分的，也是深度学习系统在实际应用中容易被攻击的原因。深度学习系统的高维输入空间巨大无法穷举，缺乏可解释性分析又造成系统逻辑分支无法划分，导致无法通过少量用例“到达”系统的所有可能场景，因此深度学习系统的完备测试目前无法实现。为了提高深度学习系统的测试充分性，发现更多缺陷，提高模型质量，相关工作者采用模糊测试方法进行深度学习系统的测试数据扩增，从而覆盖更多系统输入空间，实现测试充分性提示。同时在用例评估步骤以单条测试数据执行结果的正确与否作为评判标准，替代传统软件的异常分析方式，同样能够实现新用例的评估。因此，可以采用模糊测试方法对现有的深度学习系统进行深入评估。

Odena[19] 等人提出的 TensorFuzz 和 Guo[23] 等人提出的 DLFuzz 都是采用了上述方式对深度学习系统在 CV 领域的实践模型进行了测试。尽管核心思路相似，但在实践细节上仍有一些差异。TensorFuzz 采用基于激活层输出的覆盖率分析指导模糊测试过程，通过采用最邻近算法计算新生成的测试用例与语料集中测试用例间，输出层向量的最短距离（欧拉距离），再设置准入阈值进行筛选。而 DLFuzz 采用选择采用阈值分析单个神经元是否处于“激活状态”，并将

神经元组合的激活模式作为评估参数，如果新用例能够丰富语料集中的激活模式，则将该用例添加到语料集中。

在推荐模型测试领域，尚未有模糊测试方法的应用研究。一方面是因为推荐领域所采用的深度学习模型和计算机视觉领域的相应模型存在语义差异。另一方面，作为推荐模型输入的特征结构和数据类型更为复杂，为测试数据扩增工作带来实践难度。但相比于传统离线测试、仿真测试与线上测试流程，模糊测试方法能够以更低成本快速实现测试用例扩增，提高测试充分性，并减少测试所需时间。相比于基于对抗攻击的测试方法，模糊测试无需开发者耗费大量额外精力设计攻击流程，避免了对抗模型迭代训练造成的庞大测试成本开销问题。本系统出于对上述问题的考虑，采用模糊测试方法实现推荐模型质量评估。

2.2　常见推荐算法

推荐系统经历了漫长的技术变迁，造成了推荐模型算法和架构的多样性。作为本文实现的推荐质量评估系统所支持的测试对象，本节对常见的推荐算法进行讨论与分析，分析其内部实现原理和组织架构差异，以及其内在联系和发展。

2.2.1　协同过滤算法

协同过滤算法 [26] 是最早被提出的推荐系统算法模型，用于解决电子邮件筛选过滤问题（并非简单的垃圾邮件判断），并成为当时最主流，最有效的推荐模型，被 Amazon[27] 等公司广泛采用。协同过滤算法的核心思想是基于用户行为数据，评估不同用户的相似性或不同物品的相似性，并将相似用户感兴趣的物品进行推荐或推荐与用户感兴趣物品相似的其他物品。用户行为一般指用户与被推荐物品的交互，包括对商品的点击、收藏、购买，或者是对视频的观看、点赞、评论等。协同过滤算法有两种实现方式，分别为 UserCF[28] 和 ItemCF[29]。UserCF 算法针对 m 个用户 n 个物品，生成规模 $m * n$ 的关系矩阵，记录每名用户与每件商品间的交互情况。基于关系矩阵，采用相似度公式得到用户间相似性，以此进行物品推荐，常见相似度计算方法包括：

（1）欧氏距离。基于关系矩阵 x 中，用户 u 和 v 的行为向量（即关系矩阵中的一“行”）x_u 和 x_v，得到二者相似度 $sim(u,v)$

$$sim(u,v) = \sqrt{\sum (x_u - x_v)^2} \tag{2.1}$$

（2）余弦距离。对于用户 u 和 v 的行为向量 x_u 和 x_v，得到二者相似度 $sim(u,v)$

$$sim(u,v) = \cos(x_u, x_v) = \frac{x_u \cdot x_v}{||x_u|| \cdot ||x_v||} \tag{2.2}$$

（3）皮尔逊相关系数。对于用户 u、用户 v、商品集合 P 和集合中某一物品 p，得到用户打分 $R_{i,p}$ $R_{j,p}$ 和平均打分 $\bar{R}_i$ $\bar{R}_j$，并计算用户相似度 $sim(u,v)$

$$sim(u,v) = \frac{\sum_{p\in P}(R_{i,p}-\bar{R}_i)(R_{j,p}-\bar{R}_j)}{\sqrt{\sum_{p\in P}(R_{i,p}-\bar{R}_i)^2}\sqrt{\sum_{p\in P}(R_{j,p}-\bar{R}_j)^2}} \tag{2.3}$$

UserCF 适用于推荐物品具有时效性，物品快速更新且个性化需求较低的应用场景，如新闻推荐。而 ItemCF 更适用于物品具有稳定性（长时间不变），用户个性化需求高的场景，如电商和视频推荐等。

2.2.2　MF、LR 和 FM 算法

协同过滤算法具有良好的可解释性，但在实践过程中也暴露出了两个主要问题：（1）不具备泛化能力。协同过滤只考虑物品之间相似与否的事实，而没有其相似原因的分析，因此无法基于物品 A 与物品 B 相似的事实推出物品 C 与物品 D 相似，即协同过滤不考虑物品其他特征关系和隐含特征。由此造成的问题是严重的头部效应；（2）共现矩阵过于稀疏，对于 m 个用户 n 个物品，需要保存 $m*n$ 大小的共现矩阵，阻碍了系统的扩展。

为了解决空间占用问题，使得推荐模型具有一定的泛化能力，Koren 等人 [30] 提出 FM（矩阵分解）技术，将 $m*n$ 大小的共现矩阵转化为 $m*k$ 矩阵 A 和 $k*n$ 矩阵 B 相乘的形式，其中 k 决定了模型的泛化性能。为了得到矩阵 A 和矩阵 B，可采用奇异值分解或梯度下降法 [31]。由于奇异值分解运算复杂性较高，一般模型都选择采用梯度下降法完成矩阵分解。

尽管 FM 算法提高了模型泛化能力，降低了空间占用量，但仍然无法有效利用除交互数据外的特征信息。由此 Richardson 等人 [32] 提出 LR（逻辑回归）模型，将特征向量进行加权求和，并通过 sigmoid 函数转化为物品的 CTR（点击率）。这样简单易用、可解释性强的方法实现了对额外特征信息的有效利用，似乎“弥补了”FM 算法的缺陷。随着 LR 被广泛应用，其特征表达能力弱的问题开始暴露，由于 LR 无法将特征进行复杂交叉得到组合特征，容易造成信息理解降低的问题。为了强化特征交互，Rendle 等人 [33] 将 FM 算法的矩阵分解过程扩展到所有特征上，提出了 MF（因子分解机）算法，公式如下：

$$FM(w,x) = \sum_{i=1}^{n}\sum_{j=i+1}^{n}(w_i \cdot w_j)x_i x_j \tag{2.4}$$

其中 w 表示权重，x 表示向量。利用矩阵分解，MF 算法既有效解决了训练数据稀疏性的问题（向量组合不必出现在同一样本中），又能够利用特征交叉加深特征交互，MF 算法是 2016 年前主流推荐算法之一。

2.2.3　LS-PLM

FM 算法的提出意味着业界已经意识到特征域对于推荐模型的重要性，并产生了特征工程的雏形。然而随着 FM 算法的广泛应用，通过暴力的特征二阶交叉产生的特征关联性仍不够有效，但提高到三阶甚至更高阶，运算复杂性的指数级上升带来的资源压力又让系统难以承受。因此，何等人 [34] 提出了 GBDT+LR 的组合模型，尝试利用树模型产生更多非线性特征组合。GBDT+LR 的组合具有一定的实验意义，但在处理现实业务中频繁出现的超稀疏数据则显得乏力。为此，Gai 等人 [35] 提出了 LS-PLM 模型，通过对样本进行切片，并对每个切片采用 LR 预估，可以使模型更好地完成拟合。LS-PLM 模型是深度学习引入前最前沿的推荐系统模型，也曾是阿里巴巴业务所采用的推荐模型，相比于以往推荐算法，其优势十分明显：（1）通过可调的分片模式，LS-PLM 可以构建出丰富的特征组合模式，适应不同的业务场景；（2）模型同时采用 L1/L2 范数，参数结构稀疏性更强，更有利于模型的部署和线上服务；（3）继承了 LR 模型的可扩展性，支持分布式系统下的大规模并行训练；

2.2.4　Wide&Deep

目前已有很多针对 Wide&Deep 的改进工作，下面分别进行介绍。

针对 Wide 部分，Wang[36] 等人提出 Deep&Cross，为了强化特征交叉力度，Deep&Cross 算法使用 Cross 网络替代 Wide 部分的线性 LR 模型。相比于基于场景理解的人工特征交叉设计，自动化训练降低了设计成本，同时发现优质特征组合的潜力更大。除了采用 Cross 网络，Guo 等人 [37] 选择采用 FM 优化 Wide 部分，通过与 Deep 部分共享 Embedding 信息，一样实现了自动化特征交叉。

针对 Deep 部分，采用 FM 进行优化的思路同样可行。He 等人 [38] 提出 NFM，将 FM 算法与利用特征交叉进行性能提升的思路相结合，在 Deep 部分添加特征交叉池化层，从而将交叉结果作为深度神经网络的输入。除此之外，Xiao 等人 [39] 引入了注意力机制，提出了 AFM 模型。该模型在隐藏层后增加了注意力机制池化层，并采用权重来表征特征交叉结果的潜在注意力得分。

2.3　前端技术

本系统的可视化方案选择采用 Vue,js 渐进式框架进行前端开发，采用 ElementUI 组件库与 CSS 结合完成控件样式设计和交互反馈设计，采用 JavaScript 完成前端数据处理、数据交互和后端接口调用。

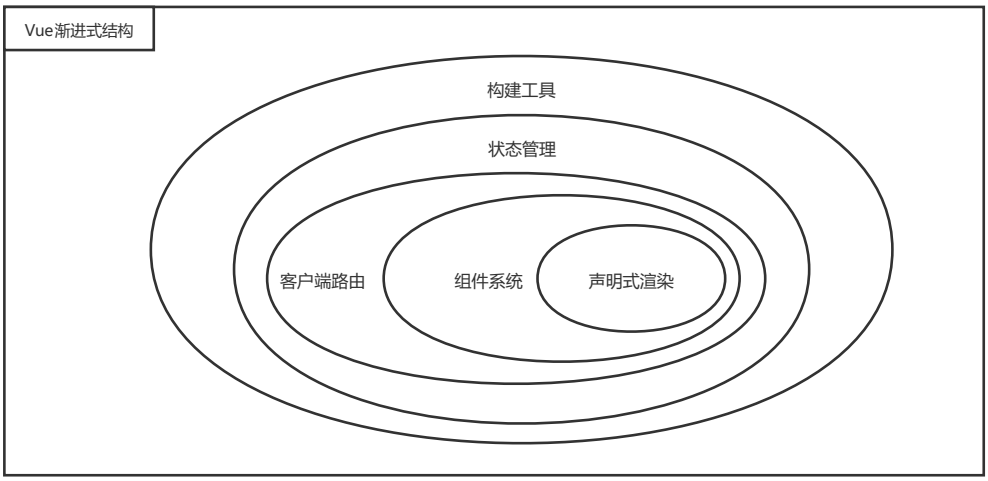

图 2.2: Vue 结构示意图

2.3.1　Vue

Vue.js [2]，是由尤雨溪开发的一款非传统的，渐进式 JavaScript 前端框架。Vue 框架基于但不完全遵循 MVVM 模式（Model—View—ViewModel），更偏向于关注状态（State) 和模板（View) 两个核心部分间的交互。Vue 通过内置的视图模板引擎完成 State 和 View 之间的数据交互，提供 vue-cli 脚手架，vue-router 等路由库等官方支持工具集，构成迭代递进，自底向上的开发生态。

Vue 的渐进式结构如图2.2所示。声明式渲染作为 Vue 的构建基础，无需复杂的 DOM 指令操作，框架提供的虚拟 DOM（Virtual DOM）自动完成状态绑定，同时减少了调用 DOM 原生 API 的性能损耗，提高了页面渲染效率。组件系统将页面结构通过组件树进行组织，每个节点都由单文件描述的组件构成。组件作为可复用的 Vue 实例，内部支持 HTML 扩展，且完整包含生命周期和钩子。客户端路由通过官方支持的管理器 vue-router 实现。在前端路由模式下，vue-router 为单个页面中的组件进行路径关联，建立 URL 和组件间的映射关系。为了在路由变更过程中进行跳转限制，vue-router 还提供了导航守卫，并细分为全局解析守卫、全局前置守卫、组件守卫、路由独享守卫等多种控制方案。状态管理将组件的状态和数据进行统一惯例，通过过 Dispatch Actions 或者直接 Commit Mutations 同步组件的状态变化，保证了多组件下的通信稳定。

相比于其他流行前端框架 AngularJS 等，Vue 提供的 API 形式更加简洁直观，同时其内部基于依赖追踪的观察机制避免了脏检查，使其在运行性能上更胜一筹，因此本系统采用 Vue.js 作为前端框架。

[2]https://cn.vuejs.org

2.3.2 ElementUI

ElementUI[3] 是阿里巴巴饿了么前端团队开源的，良好适配 Vue2.0 的桌面端组件库，其丰富的组件内容，简洁的安装使用，和完善的支持文档，为开发者、设计者和产品经理提供了便利，是目前采用 Vue 开发过程中最常用的组件库。ElementUI 的四大核心设计原则是一致性、快速反馈、高效和可控。

一致性要求组件需要与主流使用流程一致，提供组件应有的基本功能，并在整个前端系统中保证样式一致，同时组件需要遵循用户习惯的语言和概念。这意味着 ElementUI 一方面需要提供一个庞大的，功能较为完善组件库，另一方也要做好国际化；快速反馈需要满足快速页面反馈和快速控制反馈，既要使得页面元素的变化能够足够清晰地展现当前状态的变化，又需要保证用户通过界面样式和交互动态效果能够清晰地感知到自己完成的操作。这意味着 ElementUI 能够提供良好的公用组件样式设计；高效意味着简化使用组件设计不仅用户理解和使用简单，也能够帮助开发者快速上手使用；可控一方面要求用户决策可控，能够在不同场景提供清晰简洁的操作提示，另一方面要求结果可控，支持用户的撤销、回退和终止等非顺序流程操作。这就意味着 ElementUI 提供的组件能够支持提示信息的配置和内容检查，同时样式有良好的用户观感。

为此，ElementUI 提供了 55 个组件，划分为 Basic、Form、Data、Notice、Navigation 和 others6 大类，组件丰富性基本已经满足 PC 端业务的前端开发需求。ElementUI 在在 *packages*/*theme* – *chalk* 中，定义了多套公用主题样式，开发者直接在具体组件中采用主题名称进行样式设定，同时 ElementUI 也支持自定义主题，通过在线编辑器完成所有全局或组件的 Design Tokens 修改、预览和下载。为了实现国际化，良好兼容了 vue-i18n 规则，支持开发者进行自定义设置。

ElementUI 的功能强大且丰富内容，足以支持本系统的开发工作。与 BootstrapVue 等其他第三方支持 Vue 框架的 UI 组件库相比，ElementUI 更加成熟，文档完备，生态良好，同时采用原生 Vue 实现，能够更好地契合 Vue 开发工作。

2.4 后端技术

2.4.1 Django

本系统后端采用 Django 框架实现。Django[4] 是一款于 2003 年开源的，基于 Python 语言的 Web 开发框架。相比于 Flask 等其他同类 Web 开发框架，Django

[3]https://element.eleme.cn/

[4]http://www.djangoproject.com/

具有要素更齐全，工具更强大，文档、博客和社区资源更丰富等诸多优势，是一款易于维护和拓展，发展较为完善，充满活力并且发展前景光明的 Web 框架。Django 为了帮助开发者避免由于重新开发通用组件造成的重复工作和隐含缺陷，提供了大量请求响应和数据库访问等底层操作的封装接口，并使得相关复杂配置对开发者透明，因此开发者可以更加高效地完成 Web 应用逻辑实现。

表 2.1: MTV 模式

简称	全称	简介
M	模型（Model）	完成数据库 ORM（对象关系映射），支持数据表的创建，数据记录的增删改查和有效性验证。
T	模板（Template）	完成页面展现。
V	视图（View）	完成业务逻辑，包括模型的数据获取操作及模板调用，连接了 Model 和 Template。

Django 采用 MTV 模式（Model-Template-Vue）构建 Web 应用，其结构如表2.1所示。其中，模型层提供的 ORM（对象关系映射机制）能够有效支持包括 MySQL、Oracle 等多种数据库，当开发者在 Django 项目下 setttings.py 文件中完成数据库连接配置，即可采用框架内置 API 快速实现数据库访问操作，避免了手写 SQL 语句的额外开发成本。同时模型层还为开发者提供了管理员等常用对象。Django 模板层负责页面展示，原生模板引擎内置多种特殊标签简化数据交互，如“{{}}”进行变量关联，“{{}}”实现逻辑代码插入等。除此之外 Django 也支持开发者使用第三方模板引擎。Django 视图层基于 URLConf 统一管理 URL 映射关系，并支持正则表达式下的模式匹配。同时视图层还提供内置的 Request 和多种 Response 类型，如 HttpResponse、JSONResponse 和 FileResponse 等。

Django 执行流程一般包含如下几步。首先，开发者在 Django 项目下的 urls.py 文件中设定外部访问路径，并通过配置解决好跨域等访问问题后，由请求方访问指定 url。接着，Django 的请求中间件接收 Resquest 并进行数据处理，进一步交由 URLConf 完成关系映射。然后视图中间件根据映射结果调用对应方法，方法执行完成后向消息中间件发送 Response。最后消息中间件将 Response 返回请求方，由此完成了一次请求响应。由于本系统采用 Vue 框架实现页面渲染和页面交互，因此 Django 后端仅包含模型层和视图层，且前后端通过 Http 实现数据交互。Django 消息中间件接收由浏览器发送的 HttpRequest 请求，由视图层通过 URL 映射调用相应方法完成请求处理，在这一过程中通过模型层完成数据库交互和业务数据持久化，最后由消息中间件将处理结果封装为 JsonResponse 或 FileResponse 的形式返回给浏览器。

本文设计并实现的基于模糊测试的推荐模型质量分析系统，针对开发者提

供的原始数据进行较为复杂的变异处理，并完成包含矩阵降维等数据处理的质量评估。除此之外，系统还需适配推荐模型的执行环境。因此，相比于 Java 等编程语言，Python 能够更好地满足系统开发需要。作为一款流行的 Python 后端框架，Django 能够实现系统模糊测试流程的功能需要，且其强大的开源社区和详细的开发文档能够有效解决开发过程中遇到的问题。相比于 Flask 等其他 Python 后端框架，Django 提供灵活的 URL 映射规则，内置的 ORM 能够更加快捷方便地完成关系型数据库访问交互，因此本系统选择基于 Django 框架完成后端开发。

2.4.2　scikit-learn

scikit-learn[5]（scikit-learn）是一款非常流行的，基于 Python 的机器学习库，对于机器学习和深度学习领域常见的分类、回归、聚类、降维、模型选择和预处理方法有非常良好的支持。由于 sklearn 基于 NumPy，SciPy 和 Matplotlib 库构建，因此可以良好适配常见数据类型。本文选择 scikit-learn 参与推荐模型的输出结果分析和指标运算，重点用于实现精准率、召回率指标计算和 Embedding 降维。

2.5　本章小结

本章首先介绍了本系统采用的模糊测试方法，阐述了模糊测试的核心思想和基本流程，模糊测试在深度学习模型测试领域的现有工作，以及与现有推荐模型测试方法对比下模糊测试方法的优势。然后本章介绍了本系统的测试对象——推荐模型，列举了推荐模型在漫长技术演变中具有代表性的，本系统支持的模型结构和推荐算法。最后介绍了本系统开发过程中采用的相关技术。前端采用 Vue+ElementUI，保证了样式的一致性和良好的页面反馈，并通过调用后端接口进行数据交互。后端基于 Django 框架，提供了数据库访问，文件流传输和指标运算服务，指标运算过程中使用了 scikit-learn 进行降维处理。

[5]https://scikit-learn.org/stable/

第三章　系统分析与设计

本章对系统的需求分析与设计思路进行阐述。首先对系统的主要功能和工作流程进行分析。然后根据系统工作流程所需的业务层级与功能模块，给出系统的功能性和非功能性需求分析，以及系统用例图与用例描述。接着给出系统整体结构设计，阐明系统核心功能模块并采用 4+1 视图进行系统分析。最后给出系统的实体关系 E-R 图，并基于 E-R 图给出系统的持久化设计。

3.1　系统整体概述

本系统针对推荐模型进行深入的质量分析，并为开发者提供质量评估报告，减少开发者的测试工作量，帮助开发者推进模型优化。系统执行过程基于模糊测试流程，首先对开发者提供的原始数据集进行分析，并构建模糊测试语料库。然后系统采用开发者创建的特征结构，实现测试数据与特征信息的关联。接着基于相应特征类型和变异方法，完成测试数据扩增。最后系统完成评估指标计算，并为开发者提供自动化生成的测试报告。

为了提供上述功能，本系统采用 Vue+ElementUI 完成前端页面结构设计，页面提供数据集/模型文件上传，特征结构和测试任务构建，模糊测试任务的执行，任务状态信息查询，评估报告生成和扩增数据下载等可交互操作支持，并基于 Django 框架实现后端数据传输、模型执行、指标计算和数据库访问等服务。系统采用 MySQL 存储用户任务、特征结构、扩增数据和模型指标等持久化数据。

3.2　系统需求分析

3.2.1　涉众分析

本系统的涉众如表3.1所示。开发者无论是算法工程师还是测试工程师，都可以通过本系统上传测试文件，提交测试任务，查看任务进度，获取可视化测试报告，下载扩增数据。利用本系统提供的上述功能，算法工程师可以通过测试报告给出的可解释性指标寻找推荐模型的质量缺陷；测试工程师则可利用系统完成测试数据扩增和评估指标计算，通过少量代码编写和前端页面操作即可执行推荐模型测试，并获取测试报告，完成推荐模型质量分析工作。

表 3.1: 系统涉众分析

涉众	涉众需求
算法工程师	提供原始数据集，通过少量操作得到模型评估报告，获取扩增数据进行模型表现分析，重点关注并优化可解释性指标，核心诉求是帮助模型优化
测试工程师	提供原始数据集和模型文件，编写数据集分析文件，能够通过少量交互式操作完成新模型模型离线评估，重点关注通用指标，核心诉求是减少测试工作量，提高测试质量

3.2.2　系统功能性分析

基于上述系统整体概述和涉众分析部分所涉及的系统功能要素，进一步进行系统功能性分析，其相关分析结果如表3.2所示。

表 3.2: 系统功能需求列表

需求 ID	需求名称	需求描述
R1	用户创建和登录	用户能够向系统提供用户名和密码进行注册，并使用用户名和密码进行登录，并可随时登出。
R2	测试文件上传	用户能够为自动化测试任务上传原始测试数据集压缩包和模型文件压缩包，并为测试数据集提供预处理文件。同时用户还可以上传自定义特征类型文件和变异方法文件。
R3	特征结构创建	用户能够自定义测试特征结构，指明数据变异扩增过程所涉及的原始特征（通过特征名唯一标识），提供用于变异过程的变异方法和调整模式设定的变异参数，以及多个特征间可能存在的依赖关系。
R4	测试任务创建	用户能够选定已经创建的特征结构和已经上传的数据集，动态选择特征结构中的部分特征参与测试任务，设置测试规模等测试参数。
R5	测试任务查看	用户能够获取全部测试任务信息，根据任务状态筛选或测试任务名称模糊搜索定位具体测试任务，通过任务列表按钮进行报告查看和任务详情查看。
R6	评估报告生成	用户能够获取测试任务对应的测试报告
R7	扩增数据下载	用户能获取测试任务对应的扩增测试数据，并以压缩包形式下载到本地。

用户在使用本系统时，首先需要完成用户注册，提供用户名和密码并进行用户登录。用户管理对于本系统是必须的，每个用户都需要上传并依赖属于自身的测试文件，构建并查看本人私有的特征结构和测试任务，以及生成个人私有的测试报告。完成登录后，用户需要首先完成文件上传，为模糊测试过程提供原始数据。除了需要上传模型文件和其对应的原始数据集外，系统自动化测试

过程需要用户提供数据集读取文件，规定数据集预处理方式。同时，用户还可以通过文件上传页面扩展自定义特征归类和变异方法，实现测试数据扩增自定义。接着用户需要为测试任务设计特征结构，除了提供变异过程备选特征的名称和规划类型，还需要提供对应变异方法与变异参数，同时用户还应当能够规定特征间可能存在的关联性与关联方法。接着用户可以使用上传数据集、模型和特征结构创建测试任务，设定测试规模和参与变异过程的特征集合，并通过任务列表随时查看任务执行状态和任务的详细信息。当任务完成后，用户可以查看评估报告，并下载测试任务自动化生成的扩增测试数据集。

3.2.3　系统非功能性分析

考虑到本系统需要为算法工程师和测试工程师提供自动化推荐模型质量分析服务，所以本系统在满足上述功能性要求的同时，还需要根据具体使用场景的需要，满足非功能性需求，主要包括：

可靠性 由于数据集和文件类型的丰富多样，针对这些数据进行变异的过程中，很有可能因为数据格式错误，脏数据或编码问题，导致数据变异或插入过程失败，系统应当妥善处理这些问题，并抛出恰当的异常信息。同时由于创建特征结构和测试任务所需的配置较多，因此需要在用户配置错误或缺少必要信息的情况下提供清晰有效的提示和响应。

可扩展性 由于特征结构复杂，用户需求多样，内置的变异方法无法满足所有场景下用户需要。因此系统应提供特征类型和变异方法扩展接口，使用户能够添加自定义特征类型设定规则和特征变异方法。

易用性 目前完整的操作流程需要用户上传数据集和模型，创建特征结构，创建测试任务，涉及大量参数填入和传输格式，不仅容易出错，用户还可能忘记操作流程。因此需要为用户提供良好的指引，保证页面的可交互性，同时提供任务的实时运行状态和完成时间。

安全性 使用者提供的原始数据集可能包含隐私信息，因此不仅需要确保数据存储环境稳定安全，还需要降低数据透出可能，以防私密数据泄露为使用者带来不必要的损失。

性能 为了保证前端用户操作的流畅性，保证系统可用性，需要规定最长页面响应时间。文件上传成功反馈延时不高于 1 秒，特征结构创建响应时间不超过 1.5 秒，任务创建过程中获取可选数据集和特征结构下拉框的响应时间不超过 1.5 秒，创建成功后跳转响应时间不超过 1.5 秒，下载响应不超过 1 秒。

3.2.4　系统用例图

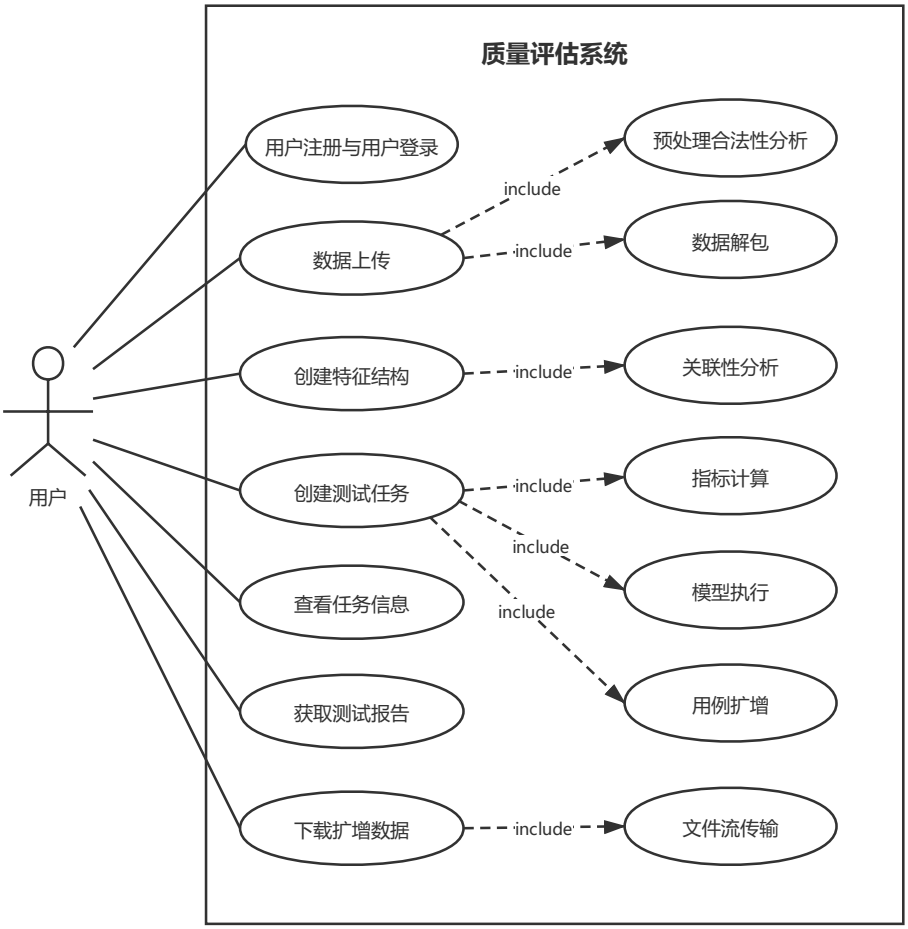

图 3.1: 系统用例图

在完成系统功能性分析与非功能性分析后，生成如图3.1所示的推荐模型质量分析系统用例图。用户可以在系统上完成用户注册与用户登录，上传数据集与模型文件，创建特征结构与测试任务，查看任务进度与测试结果，下载扩增数据。其中，用户每次上传数据集与模型文件，都会触发合法性检查，如果是压缩文件则进行解包与数据库持久化记录添加，而如果是数据集预处理文件则进行预处理合法性分析。系统对用户上传的特征结构执行关联性分析，将可视化表示下的特征连线表示转化为持久化数据中对应记录的外键。用户创建测试任务成功后将会自动依次执行测试用例变异扩增，用例模型执行，基于模型执行结果的指标计算等步骤。用户除了能够获取测试报告，还能下载系统自动化生成的扩增数据，通过文件流传输模块从服务端下载到本地。

用例图中的用户注册与登录用例、数据上传用例、创建特征结构用例、创建

测试任务用例、查看任务信息用例、获取测试报告用例和下载扩增数据用例为主要用例，下面针对这批主要用例给出详细用例描述。

表 3.3: 用户注册与登录用例描述

ID	UC1
名称	用户注册与登录
参与者	参与者：平台使用者 目标： 1. 实现新用户注册 2. 用户系统登录
触发条件	用户点击“登录”
前置条件	无
后置条件	无
优先级	高
正常流程	1. 用户输入用户名与密码 2. 用户点击“登录”按钮跳转
异常流程	2a. 如果用户已存在但密码不正确，提示密码错误 2b. 如果新用户用户名包含异常字符，提示用户名内容不合法 2c. 如果新用户用户名超过 15 个字符，提示用户名过长 2d. 如果新用户密码包含非数字或大小写字母，提示密码不合法 2e. 如果新用户密码少于 8 个字符，提示密码过短 2f. 如果新用户密码多余 20 个字符，提示密码过长

表3.3给出了用户注册与登录用例描述。在登录页面，用户在输入框中输入用户名和密码，如果用户存在且密码正确，点击“登录”按钮即可完成登录并跳转到系统主页面，否则提示用户密码错误且不进行跳转。如果用户不存在，则直接新建并保存新用户信息，同时完成主页面跳转。其中需要对用户名和密码进行限制。用户名最长 15 个字符，且只能由英文大小写字母、数字和汉字组成，格式无具体要求。密码最短 8 个字符，最长 20 个字符，必须且只能包含数字和英文大小写字母。跳转后的主页面展示了用户测试任务相关的统计信息。包括 7 天内任务成功数、失败数、正在执行中数以及全部任务的成功数、失败数和正在执行中的任务个数。除此之外，主页面还给出反映用户 7 天内任务成功数、失败数等数据的折线图。

表3.4给出了创建特征结构的用例描述。当用户完成登录后，即可点击侧边栏的“创建特征结构”按钮进入特征结构组件页面。首先，用户需要提供待新建特征结构的结构名称，该名称用于在该用户所创建的所有特征结构中，为每个特征结构进行唯一标识，如果存在重复特征名称，则该特征结构无法被创建。同时，特征结构名称只能由汉字、大小写英文字母、数字和下划线组成，且特征结构名称不能超过 20 个字符。当完成特征名称填入后，用户通过鼠标右击特征

表 3.4: 创建特征结构用例描述

ID	UC2
名称	创建特征结构
参与者	参与者：平台使用者 目标： 1. 利用特征名、特征类型、变异方法和变异参数等信息，创建特征单体 2. 基于特征单体，创建特征结构
触发条件	用户点击“创建特征结构”
前置条件	用户完成登录
后置条件	数据库新增特征结构记录
优先级	高
正常流程	1. 用户在侧边栏点击“新建特征结构”，进入结构组建页面 2. 用户输入特征结构名称 3. 用户根据需求特征个数，新建特征块 4. 用户为每个特征块填写相关参数 5. 用户创建特征关联关系 6. 用户点击“立即创建”
异常流程	2a. 如果用户未输入特征结构名称，则填入后直接提示“结构名称未输入” 2b. 如果特征结构名称超过 20 个字符，填入后直接提示“结构名称过长” 2c. 如果特征结构名称包含不合法字符，填入后直接提示“不合法的结构名称” 2d. 如果用户输入结构名称已存在，则点击“立即创建”按钮后无法完成结构创建，并显示结构名称已存在的错误提示 4a. 存在特征信息不全的特征单体，则点击“立即创建”按钮后无法完成结构创建，并显示特征单体信息缺失 4b. 存在特征名重复的特征单体，则点击“立即创建”按钮后无法完成结构创建，并显示特征单体信息错误

结构空间空白位置，或在空间下方选择特征结构并点击添加按钮，完成特征单体创建。每个特征单体以特征块的形式表现。单体创建完成后，左键单击每个特征块，在结构空间左上角填写该特征对应的特征参数，包括特征名、特征类型、变异方法、变异参数和依赖文件名等。如果需要进行特征关联测试，且构建的特征单体集合中存在特征关联关系，用户可以按照依赖关系将相关特征进行连接。最后用户点击“创建”按钮，触发针对结构名称和特征单体信息规则检查。如果特征名称在当前用户创建的结构集合中未存在，所有特征单体信息完整且不存在特征名称重复等信息错误，则成功完成特征结构创建，同时结构信息将被服务端转化为对应的持久化记录存储到数据库中。

表3.5给出了创建测试任务的用例描述。当用户完成登录并进入测试任务创

表 3.5: 创建测试任务用例描述

ID	UC3
名称	创建测试任务
参与者	参与者：平台使用者 目标： 创建推荐模型模糊测试任务
触发条件	用户点击“创建测试任务”
前置条件	1. 用户完成登录 2. 用户完成数据集、模型文件、特征结构创建
后置条件	数据库新增任务记录，执行测试流程
优先级	高
正常流程	1. 用户填写测试任务名称 2. 用户选择测试模型文件 3. 用户选择测试数据集 4. 用户选择测试过程采用的特征结构 5. 用户根据特征结构选择本次测试待变异特征 6. 用户设定测试任务的测试规模、抽样规模和变异规模 7. 用户选择是否采用关联模式进行测试 8. 用户点击“立即创建”按钮，提交测试任务
异常流程	1a. 如果用户未输入测试任务名称，则填入后直接提示 “任务名称未输入” 1b. 如果测试任务名称超过 20 个字符，填入后直接提示 “任务名称过长” 1c. 如果测试任务名称包含不合法字符，填入后直接提示 “不合法的任务名称” 1d. 如果用户输入测试任务名称已存在，则点击“立即创建”按钮 可以完成测试任务提交，但显示任务名称已存在的警告 3a. 如果用户选择的模型文件和数据集不对应，则点击 “立即创建”按钮后提示“模型与数据集不匹配” 6a. 用户未设定参数值，直接提示相应参数未输入

建页面后，首先需要填写测试任务名称。测试任务名称只能由由汉字、大小写英文字母、数字、下划线和短横线组成，且任务名称不能超过 20 个字符。任务名称可重复，但系统并不推荐，因此在提交过程中会给出警告提示。完成测试任务名称确认后，用户需要依次选择测试任务对应的待测模型，原始数据集和特征结构，并且系统会检查待测模型和原始数据集的匹配性。选定特征结构后，系统将会解析出所有相关的特征单体，并以特征单体名称复选框组合的形式展现给用户。用户可以自由地选择特征结构中的特征子集进行测试。在完成待测特征集合设定后，用户还需要确定测试规模、抽样规模和变异规模，用于控制测试任务的变异方式和扩增数据规模。其中测试规模用于设定最终产出的扩增用例条数，抽样规模表示每次迭代从原始数据集中抽取的用例条数，变异规模

表示基于单条抽取的原始用例生成的新用例条数。如果上述数据填写完整无误，用户点击” 立即创建 “按钮即可提交测试任务并开始模糊测试流程，否则页面将给出相应的错误提示或警告信息。

表 3.6: 上传数据用例描述

ID	UC4
名称	上传数据
参与者	参与者：平台使用者 目标： 1. 获取用户上传的模型文件和数据集文件，保存到本地 2. 获取用户上传的数据集预处理文件，并完成绑定 3. 获取用户上传的特征划分文件和变异方法文件，并导入系统
触发条件	用户以拖动或浏览的方式上传文件
前置条件	用户完成登录
后置条件	文件存储到本地，添加相应持久化记录
优先级	高
正常流程	1. 用户通过拖拽或浏览本地文件方式选择文件上传 2. 页面提供上传成功反馈
异常流程	1a. 上传数据集或模型文件非 zip 格式，则上传失败 1b. 上传预处理文件或自定义文件非.py 文件，则上传失败 1c. 上传预处理文件命名不合法，则上传失败

表3.6给出了上传数据的用例描述。当用户完成登录后，即可上传测试任务所需的原始数据集文件、模型文件、数据集预处理文件和自定义功能扩展文件。用户上传的模型文件必须为 zip 压缩包形式，当上传成功并解压无误后，系统将本地保存模型文件，并存储模型的持久化记录。用户上传的数据集文件必须为 zip 压缩包形式，但当上传成功并解压无误后，系统只本地保存解压后的数据集相关文件，除非已存在相应的数据集预处理文件。用户上传的数据集预处理文件必须为.py 格式，且命名必须满足系统约定。如果已存在相应的数据集，则完成匹配并添加测试可选数据集持久化记录。无论是否匹配成功，系统都保存该预处理文件到服务器本地。上传过程不采取文件内部格式审查和可执行检查，因此用户上传的自定义特征类型和变异方法将直接存储到服务器本地相应位置。

表3.7给出了测试任务查看的用例描述。用户完成登录后即可查看当前用户已经成功提交的所有测试任务。由于测试任务较多，用户可以使用系统提供的任务查找功能，通过任务状态筛选器和任务名称搜索框来查找目标任务记录。一条任务记录包含任务基本信息与运行状态，具体来说包含了任务的持久化记录 ID、任务名称、提交时间、完成时间、任务状态和数据规模。其中任务状态包含 “running” 正在执行中，“success” 已完成和 “failed” 失败三种状态。如果处于 “running” 状态，则完成时间为空，否则为任务执行结束时间。数据规模由用户

表 3.7: 查看任务信息用例描述

ID	UC5
名称	查看任务信息
参与者	参与者：平台使用者 目标： 1. 用户可以获取包含本人创建全部测试任务的任务列表 2. 用户可以筛选任务列表 3. 用户可以通过任务列表查看任务详情和测试报告
触发条件	用户查看任务列表
前置条件	用户完成登录
后置条件	无
优先级	高
正常流程	1. 用户点击“测试任务列表” 2. 用户使用筛选，搜索或翻页找到对应的测试任务 3. 用户查看任务执行状态，进行相关操作
异常流程	2a. 如果搜索框为空，点击搜索直接重置当前任务列表 3a. 任务处于“未完成”状态，则无法查看测试报告

提交测试任务时的相关参数决定。无论任务处于哪种状态，都可以点击“详情”按钮获取任务详细信息，包括任务所用的数据集、模型和测试结构等信息。如果任务未处于“success”状态，则无法跳转到测试报告页面。

表 3.8: 获取测试报告用例描述

ID	UC6
名称	获取测试报告
参与者	参与者：平台使用者 目标： 查看测试任务对应的测试报告
触发条件	用户查看任务列表过程中，点击“报告”按钮
前置条件	1. 用户完成登录 2. 任务处于已完成状态
后置条件	无
优先级	高
正常流程	1. 用户点击“测试任务列表” 2. 用户使用筛选，搜索或翻页找到对应的测试任务 3. 用户点击“报告”按钮 4. 等待报告页面跳转和渲染
异常流程	3a. 任务不处于“已完成”状态，则点击无效 4a. 部分指标计算失败，则该部分指标不显示

表3.8给出了获取测试报告的用例描述。用户首先从任务列表中找到对应的测试任务，且任务处于已完成状态，然后点击“报告”按钮跳转到报告展示页

面。报告展示页面将给出所有已经完成计算的指标数据，形式包括量化值和图表。如果部分指标还未完成计算或计算异常，则不显示这部分指标。

表 3.9: 下载扩增数据用例描述

ID	UC7
名称	下载扩增数据
参与者	参与者：平台使用者 目标： 获取扩增测试数据，方便参与者进行后续分析
触发条件	用户点击页面中任务记录的“下载”按钮
前置条件	1. 用户完成登录 2. 任务处于”已完成“状态
后置条件	无
优先级	高
正常流程	1. 用户点击侧边栏的”下载扩增数据” 2. 用户使用搜索或翻页找到对应的测试任务 3. 用户点击“下载”按钮
异常流程	2a. 如果搜索框为空，点击搜索直接重置当前任务列表

表3.9给出了下载扩增数据的用例描述。用户在查看任务列表过程中不提供下载功能，主要因为只有成功执行的任务才能进行下载。用户点击侧边栏的”下载扩增数据”获取可下载的任务列表。用户可通过与查看任务列表页面相似的操作，利用搜索框完成测试任务查找。用户点击“下载”按钮即可将该条测试任务对应的扩增数据下载到本地。用户也可点击“删除”按钮删除数据。

3.2.5　系统流程图

基于上述用例详细描述，给出用户使用的系统流程图，如图3.2所示。

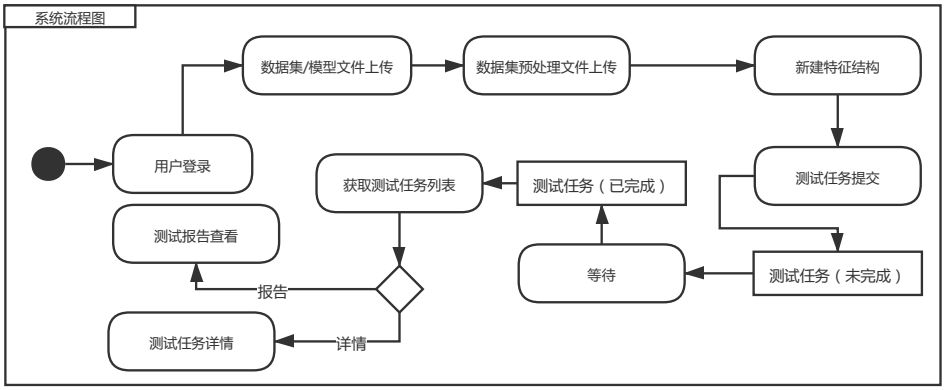

图 3.2: 系统流程图

3.3　系统总体设计

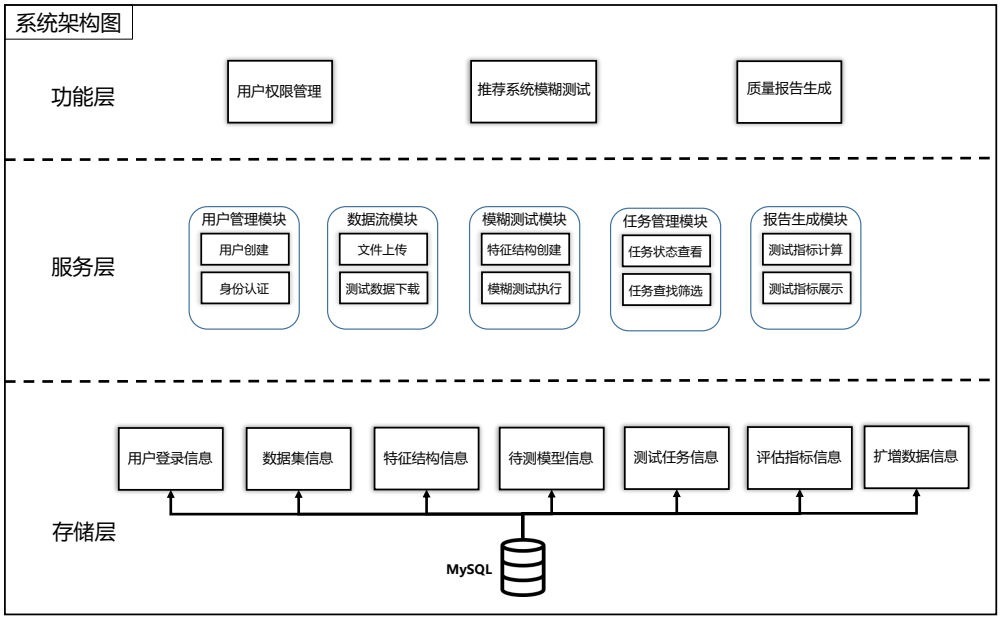

图 3.3: 系统架构图

系统整体架构如图3.3所示。从用户、业务逻辑和数据三个角度，将系统拆解为三层，分别为用户体感可达的功能层，为功能层提供业务逻辑支持的服务层，以及为服务层保留和持久化数据的存储层。本系统基于 B/S 架构，采用 MTV（Model-Template-Vue）模式开发。其中，功能层为用户提供了核心服务的数据展示和交互入口，采用 Vue+JavaScript+ElementUI+CSS 完成静态页面设计和页面交互设计。通过 axios 向服务器发送 GET/POST 请求，服务器完成数据处理后构建并向浏览器返回 JSONResponse 或 FileResponse。根据业务逻辑功能进行划分，服务层由用户管理模块，数据传输流模块，模糊测试模块，任务管理模块和报告生成模块构成，所有业务逻辑采用 Django 框架实现，依赖于 Django 架构丰富的通用组件和 URLConf 映射管理机制，保证了业务逻辑的快速迭代。存储层采用 MySQL 数据库保留持久化数据，利用 Django 提供了统一的数据库访问接口。测试数据集和模型文件等非结构化数据保存在本地，其余数据保留在数据库中。

3.3.1　模块说明

下面针对整体架构图3.3中各层下不同模块进行说明。

功能层即前端用户可感的，为用户直接提供的服务内容。用户权限管理除了向用户提供系统注册和登录服务，还保证了不同用户间的数据集、模型文件、特征结构和测试任务信息不共享。推荐模型模糊测试是核心功能，包括了系统功能需求中所涉及的文件上传、特征结构创建、测试任务创建、测试任务查看和测试数据下载等功能。当用户完成数据集和模型相关文件上传，创建特征结构并成功提交测试任务后，服务器将会依次完成测试用例变异扩增，模型执行，执行结果收集和指标计算等步骤。质量报告生成功能将最终计算得到的指标数据整理并渲染，向用户展现包含量化指标和可解释性图表的测试质量报告。

逻辑层根据后端实现分为不同模块。用户管理模块负责与用户数据表交互，包括新用户注册，用户列表获取，用户密码检查等子功能。数据流模块负责数据上传下载，包括文件上传，文件解压和命名合法检查，扩增数据文件下载等功能。模糊测试模块负责特征结构创建和测试任务创建。为了支持特征结构创建，后端提供了特征关联分析和特征结构持久化等功能。为了支持测试任务创建，后端提供了数据集分析，模型加载和特征结构分析，并为模糊测试任务的执行提供测试用例变异扩增，扩增数据执行和指标计算等功能。任务管理模块负责为用户提供测试任务相关信息，包括任务执行状态和完成时间等，并提供任务查找筛选和报告查看等功能。报告生成模块负责读取持久化指标数据并进行分析整合，再将处理结果提供给功能层实现报告渲染。

存储层持久化针对系统内部的实体关系构建数据表，主要包括用户登录数据表、数据集信息数据表、特征结构数据表、模型数据表、测试任务数据表、评估结果数据表、图表显示数据表和扩增文件数据表等，具体表结构将在本章持久化部分进行详细分析。

3.3.2　4+1 视图

Kruchten 等 [40] 提出的“4+1”视图概念，能够产生自上而下的系统设计拆解，并对系统进行全面合理的抽象描述。4+1 视图包括逻辑视图、开发视图、进程视图、物理部署视图和场景视图。由于场景视图已在本章 3.2 节中用例分析部分进行阐述，故下面将对剩余 4 个视图进行分析。

逻辑视图　如图3.4所示，逻辑视图是系统职责的逐级划分，并描述各逻辑元素间的关系。由于逻辑视图主要从用户的角度出发进行构建，因此面向对象的编程语言可以采用 UML 类图作为逻辑视图的具体表示。其中，RSTestService 是本系统的核心，提供了文件上传服务 FileUploadService、特征结构创建服务 FeatureStructureService 和测试任务服务 TestTaskService。FileUploadService 负责实现用户文件上传，包括用户数据集文件、用户模型文件、用户数据集预处理

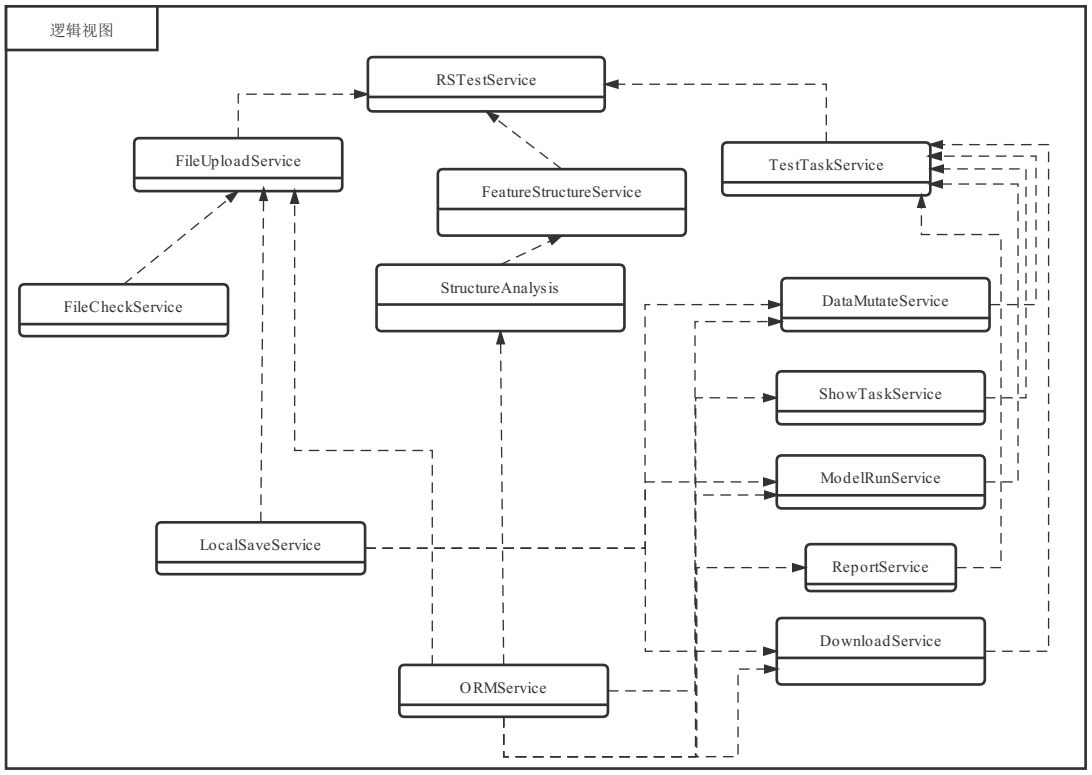

图 3.4: 逻辑视图

文件和用户自定义功能扩展文件。除了提供文件数据流传输服务外，FileUploadService 还依赖其他服务。其中 FileCheckService 负责上传文件命名和内容的合法性检查，LocalSaveService 负责将文件保存到服务器本地，ORMService 完成与数据库交互。特征构建服务 FeatureStructureService 主要负责完成前端发送的特征 JSON 数据解析，并转化为特征数据记录存储到数据库中。其依赖的 StructureAnalysis 负责特征关联性分析，同时通过 ORMService 实现数据库交互。测试任务服务 TestTaskService 本身除了负责接收数据并初始化测试任务对象外，还依赖于多个子服务实现测试流程，每个子服务都通过 ORMService 实现数据库交互。数据变异扩增服务 DataMutateService 负责完成测试用例扩增流程，除了通过 ORMService 完成任务状态更新和本地下载记录添加外，还依赖 LocalSaveService 完成扩增数据的本地化存储。ShowTaskService 负责向前端提供测试任务列表和任务详情信息，包括当前任务的状态、任务的运行时间和任务的测试规模等。ModelRunService 负责完成模型执行与执行数据收集。ReportService 主要负责实现指标计算，并将指标数据发送给前端。DownloadService 负责向前端提供扩增数据的下载列表，以及响应前端发送的数据下载请求和数据删除请求。

开发视图 如图3.5所示，开发视图是系统的架构描述，分析了系统不同部分

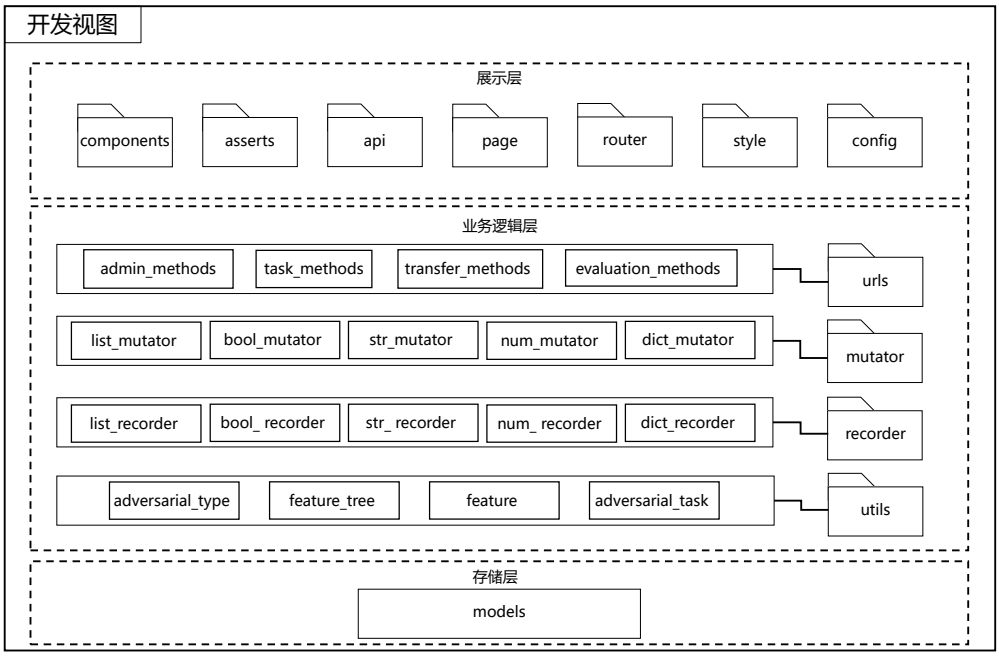

图 3.5: 开发视图

的文件组织管理方式。本系统采用前后端分离架构。

展示层即系统的前端部分，主要包含 componets、asserts、api、page、router、style、config 7 个文件夹。components 文件夹中保存的是组件的.vue 格式文件，这些组件将会在整个前端系统中被多次复用，其中具体包括组件为页眉组件、侧边栏组件、图表组件、特征块组件等。asserts 文件夹中包含静态资源文件，包括展示用图片和 JSON 配置等。api 文件夹中存放获取后端接口的接口映射规则和调用方法，具体来说，api.js 存储了后端 GET/POST 接口的路径名称，http.js 设定了与后端交互的 get/post/delete/upload 方法的具体实现。page 文件夹包含了整个前端系统的核心部分——页面展示的.vue 文件，每个页面都由一个独立的.vue 文件组成。router 文件夹包含了 index.js 文件，用于配置系统前端的路由规则。style 包含了系统的全局样式。config 包含了系统的全局配置。

逻辑业务层包含了系统的后端部分。主要包含 urls、mutator、recorder、utils 4 个文件夹。urls 负责向前端提供数据接口，包括登录接口、文件流接口、任务和特征结构创建接口、评估指标数据获取接口等，并完成 URL 映射。mutator 和 recorder 分别保存了系统内置的变异方法和记录器结构，用户可以直接使用。utils 则存放所有中间类和方法，用于支持接口的功能实现和模糊测试流程。

存储层与 MySQL 数据库交互直接依赖 Django 强大的 ORM 机制，通过在 models.py 文件中完成表结构定义，即可采用内置 API 完成数据库的增删改查操

作，无需额外编写 DAO 文件或用于数据交互的 sql 语句。

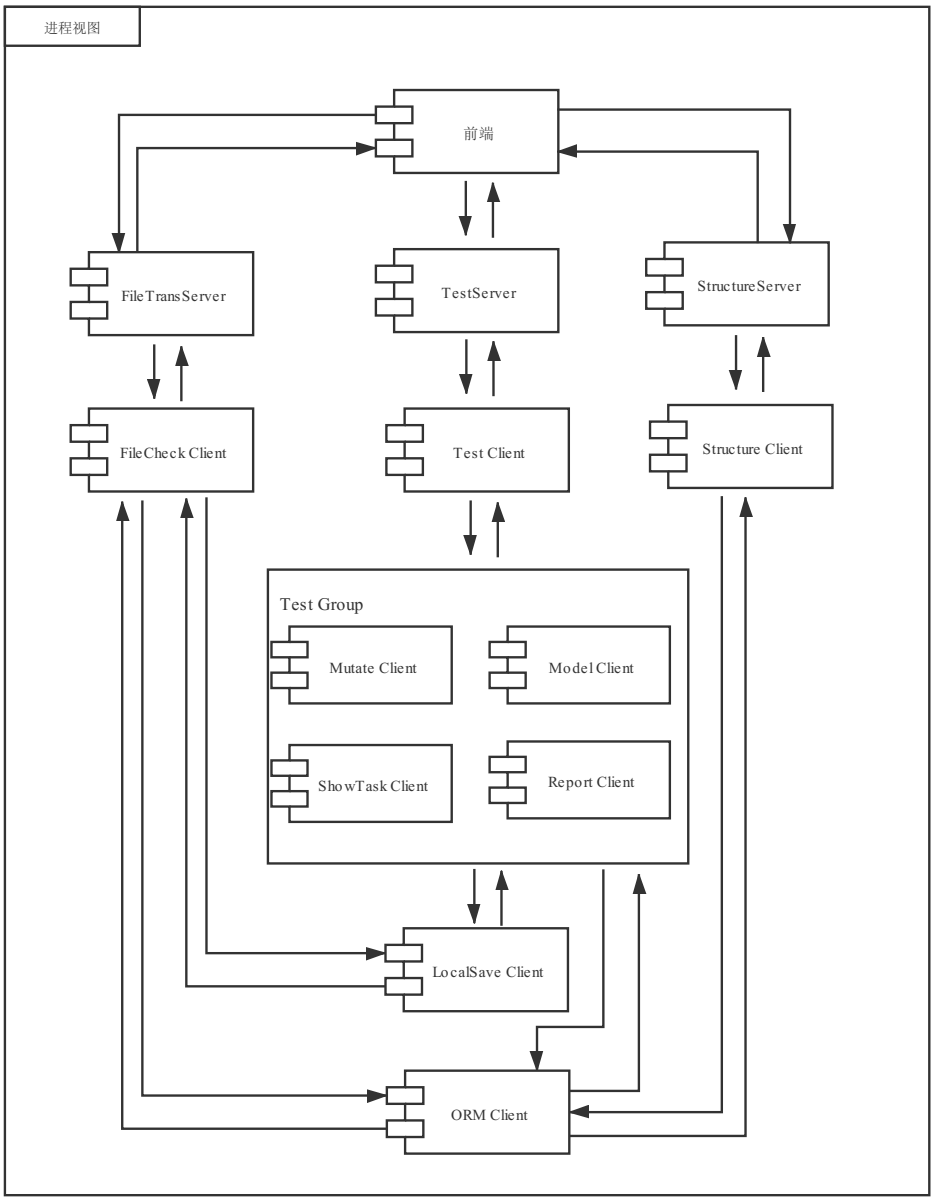

图 3.6: 进程视图

进程视图 如图3.6所示，进程视图清晰表述了在用户交互过程中，系统各部分的通信方式。当用户在操作的过程中，前端通过与文件上传服务、特征结构服务和推荐模型模糊测试服务交互，实现数据集与模型相关文件上传，特征结构创建，测试任务创建与执行，报告生成等功能。其中，文件传输服务（FileTransServer）处理上传请求时首先与本地存储服务交互，完成用户上传文件的本地存储，然后执行文件检查完成数据集匹配或将不合法文件从本地移除，最后与数据库交互完成持久化数据存储。处理下载服务则先通过 ORM 获取本地文件路径，再进

行本地文件存在性检查，最后返回文件流。当前端向服务器请求添加新特征结构时，特征结构服务（StructureServer）首先进行特征结构分析，再将分析结果转化为多条特征记录，通过调用 ORM 方法存储到数据库中。当前端需要提交测试任务，查看测试任务列表或是查看测试报告时，都需要与推荐模型模糊测试主服务进行交互。当提交新测试任务后，主服务（TestServer）首先通过 Mutate Client 进行测试任务对象初始化与测试用例变异扩增，再由本地存储服务将扩增数据存储到本地。接着通过 Model Client 执行模型，并通过 Report Client 收集的执行结果计算评估指标，并将评估指标通过 ORM 存储到数据库中，最后向前端返回 Response。如果前端向服务器请求获取测试任务列表或生成报告，服务器通过 ShowTask Client 访问持久化数据返回相应结果。

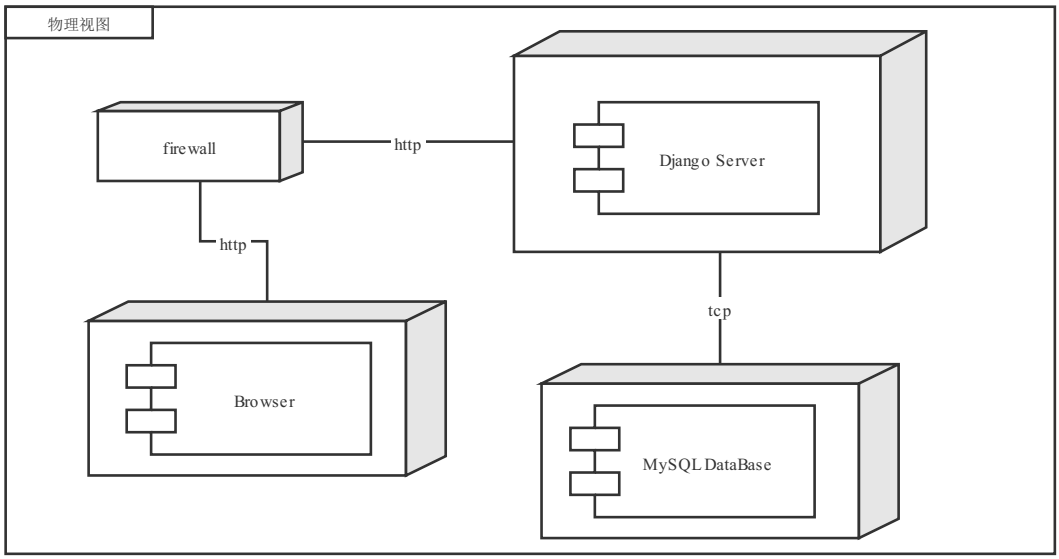

图 3.7: 物理视图

物理视图 如图3.7所示，物理视图向运维人员表述了如何将系统完成部署，以及部署后系统的真实运行方式。由于本系统未用到微服务模式，后端仅需一个 Django 服务器提供单服务，一个 Mysql 服务器。Django 服务器接收前端 HTTP 请求，与数据库交互，向前端发送 HttpResponse，从而实现前后端分离。

3.4　持久化设计

本系统的持久化设计涵盖系统核心功能所依赖的结构化数据，包括用户信息、测试可选的数据集与模型信息、特征结构信息、测试数据信息、评估指标信息和文件下载信息。图3.8给出了持久化设计中的系统实体关系，其中每个用户

实体都对应多个数据集、模型、特征结构和测试任务实体。测试任务实体与数据集、模型、用户、扩增数据、特征结构和评估结果实体一一对应。由于评估结果涉及图表，因此一条评估结果实体与多个散点实体对应。下面给出每个实体对应的数据库表结构，并一一说明。

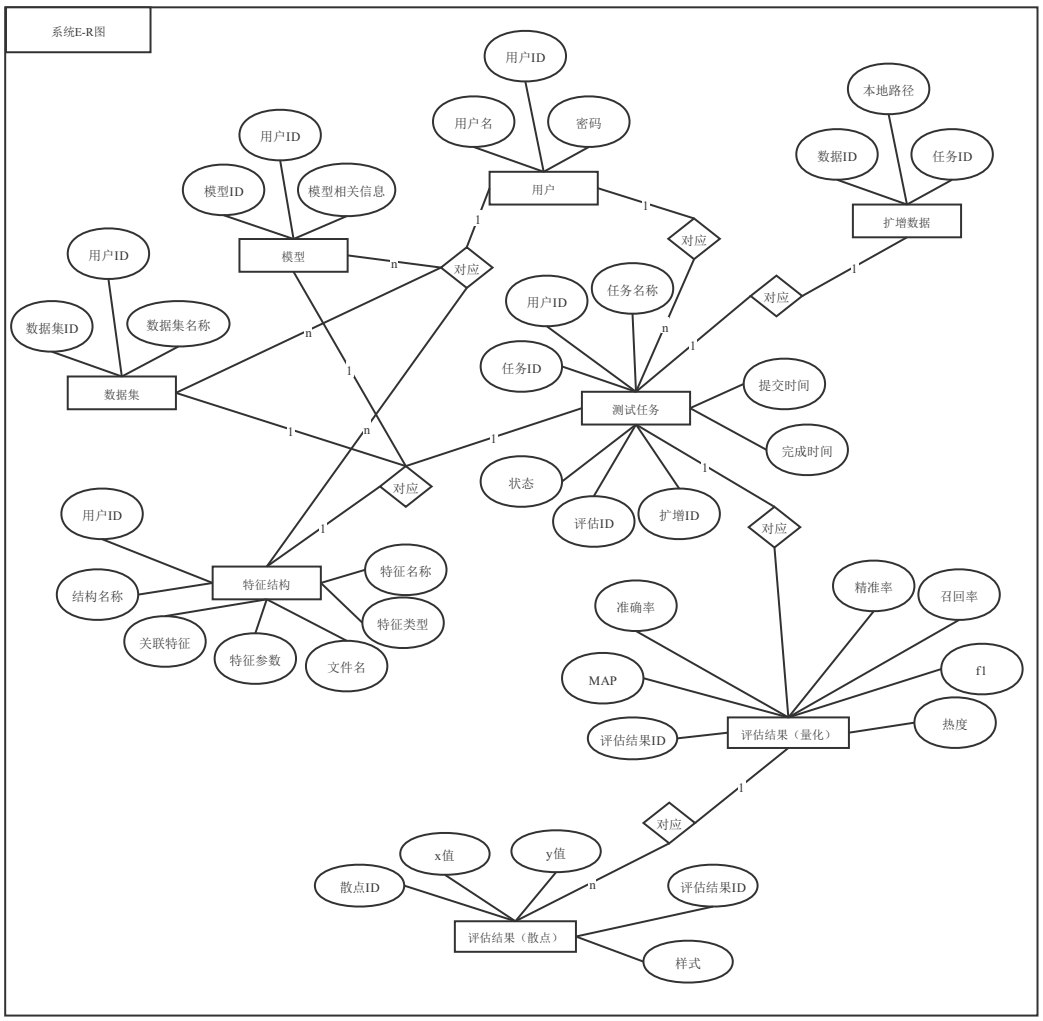

图 3.8: 系统 E-R 图

用户信息表如表3.10所示。除了采用 id 作为用户唯一标识，还保存用户名称 user_name 和登陆密码 password，用于实现登录功能和用户信息展示。

模型文件表如表3.11所示，id 用于模型唯一标识，user_id 完成模型实体与用户实体关联，model_name 用于方便测试任务创建过程中用户进行模型选择，related_dataset 用于实现测试任务创建前数据集模型匹配检查，model_type 影响模型执行方式。

测试任务表如表3.12所示，user_id 完成与用户实体关联。task_name、dataset_name、

表 3.10: 用户信息表

字段	含义	类型	说明
id	用户 ID	int	主键
user_name	用户名	varchar	用户名，保证唯一性
password	密码	varchar	用户密码

表 3.11: 模型文件表

字段	含义	类型	说明
id	模型 ID	int	主键
user_id	用户 ID	int	所属用户，外键
model_name	模型名	varchar	模型名称
related_dataset	关联数据集	int	模型采用的数据集 ID 集合，如有多个采用, 分隔
model_type	模型类型	varchar	记录模型类型

model_name、structure_name 分别对应测试任务名称和测试任务依赖的数据集、模型、特征结构。submit_time 与 finish_time 用于记录任务的开始时间和完成时间，完成时间包含成功完成时间和失败时间。state 标识任务当前处于“success”、“running”、“failed”三种状态之一。number 用于记录测试规模，result 则与评估结果实体相关联。args 记录了其他任务参数，包括单条抽样个数等信息。

表 3.12: 测试任务表

字段	含义	类型	说明
id	测试任务 ID	int	主键
user_id	用户 ID	int	所属用户，外键
task_name	任务名	varchar	测试任务名称
dataset_name	数据集名	varchar	数据集名称
model_name	模型名	varchar	模型名称
structure_name	结构名称	varchar	特征结构名称
submit_time	提交时间	datetime	测试任务的提交时间
finish_time	完成时间	datetime	测试任务的完成时间
state	任务状态	varchar	指定任务当前状态
number	规模	int	指定测试任务扩增用例数量
result	关联指标 ID	int	相关指标 ID
args	参数	json	测试任务所需额外参数

特征结构表如表3.13所示，user_id 完成特征结构实体与用户实体关联。一个完整的特征结构被拆分为多条记录，每条记录对应特征结构中的一个特征单体，特征单体由 feature_name 特征名表示，从属关系 structure_name 确定。feature_types、adversarial_args、adversarial_func、file_name 与用例描述一致，分别记录特征类型、特征变异参数、特征变异方法和特征所属的数据集文件名。re-

lated_ids 保留关联特征记录 ID 集合来反映特征间的关联性。

表 3.13: 特征结构表

字段	含义	类型	说明
id	特征结构 ID	int	主键
user_id	用户 ID	int	所属用户，外键
structure_name	结构名称	varchar	特征结构名称
feature_name	数据集名	varchar	特征名称
feature_types	特征类型	varchar	特征所属具体类型，主要用于后端自动绑定统计器
adversarial_args	模糊测试变异参数	json	用于初始化特征统计器和执行模糊测试变异方法
adversarial_func	模糊测试变异方法	varchar	指定针对该特征采用的变异方法名称
related_ids	关联特征	varchar	指定相关特征 ID 集合，用, 分隔
file_name	关联文件名	varchar	特征相关的数据集文件名

数据集表如表3.14所示，user_id 完成数据集实体与用户实体关联，dataset_name 方便用户创建测试任务时进行选择，dataset_dir 用于数据预处理和扩增过程中进行本地数据集文件访问。

表 3.14: 数据集表

字段	含义	类型	说明
id	数据集 ID	int	主键
user_id	用户 ID	int	所属用户，外键
dataset_name	数据集名	varchar	数据集名称
dataset_dir	本地路径	varchar	记录数据集存储的本地路径

评估结果表如表3.15所示。task_id 完成与测试任务实体关联，其他字段记录了相应量化指标结果。评估结果表不包含绘图所用的散点数据。

表 3.15: 评估结果表

字段	含义	类型	说明
id	评估结果 ID	int	主键
task_id	任务 ID	int	所属任务，外键
accuracy	准确率	decimal	模型准确率
precision	精准率	decimal	模型精准率
recall	召回率	decimal	模型召回率
f1	f1	decimal	模型 f1 值
MAP	MAP	decimal	模型平均精度均值
hot	热度值	decimal	模型热度值
hot_cols	热度分桶	varchar	模型热度分桶

评估散点表如表3.16所示。散点表记录了绘图所用的散点信息，evaluation_id 用于与评估指标记录进行关联，x_val 和 y_val 用于点定位，figure 用于在绘制过

程中指定图表样式，目前支持散点图“scatter”和折线图“line”。label 用于绘制过程中确定点的标签信息，如点所属的商品类型等。

表 3.16: 评估散点表

字段	含义	类型	说明
id	评估散点 ID	int	主键
user_id	用户 ID	varchar	所属用户，外键
x_val	横坐标值	double	散点横坐标
y_val	纵坐标值	double	散点纵坐标
evaluation_id	评估结果 ID	int	所属评估结果，外键
figure	模式	varchar	散点对应的表形式
label	标签	varchar	散点对应的标签值

扩增数据下载表如表3.17所示。task_id 完成与测试任务实体关联，dir 记录扩增数据的服务器本地路径。

表 3.17: 扩增数据下载表

字段	含义	类型	说明
id	下载记录 ID	int	主键
task_id	用户 ID	varchar	所属任务，外键
dir	路径	varchar	扩展数据本地路径

3.5　本章小结

本章分析并给出了基于模糊测试的推荐模型质量分析系统的整体设计。系统整体概述部分阐明了本系统的设计目标和主要功能，以及完成功能实现所使用的主要技术。系统需求分析部分从涉众分析出发，明确了系统的目标用户期望，基于用户期望给出了用户创建和登录、测试文件上传、测试任务创建等七个系统功能性需求，并给出保证可用性等系统指标的非功能性需求。接着具体化功能性需求，将功能性需求转化为系统用例图，并给出用例图中关键用例的详细描述，最后从用户角度给出系统的使用流程图。系统总体设计部分在需求分析的基础上进行系统结构分析，并对系统不同层面的模块功能进行更细致地划分，并给出 4+1 视图从不同视角给出更全面的系统抽象描述。系统持久化设计部分给出系统实体关系，并基于实体关系给出持久化数据表结构并进行说明。

第四章　变异方法与评估指标

本章主要介绍系统内置的特征类型，与特征类型相关或与数据集相关的变异方法，以及系统内置的评估指标。内置特征主要分为 5 个大类，分别为数值型、字符型、布尔型、列表型和字典型。对于每种类型下的每个子类型特征，系统内置一到两种变异方法。除此之外，系统还提供与具体特征无关的，数据集层面的变异方法。系统内置的评估指标包含量化指标与图表两种形式。

4.1　特征分类设计

开发者上传的原始数据集结构复杂，特征种类与记录形式多样，系统无法自动化完成测试数据属性值所属特征类型分析，比如说当处理用户生日特征时，无法针对 “19970824” 这个值去分析它是时间特征还是整型特征。可以说，特征类型是由特征值形式和特征语义共同决定的，由于构建自动化格式解析和语义分析，或构建特征类型实时推荐服务的工程量过大，因此本系统需要使用者进行手动设置。为了方便使用者特征选型，减少其操作复杂性，本系统内置了数值型、字符型、布尔型、列表型和字典型 5 种特征类型，每个类型下有一到多个子类型，足以覆盖大多数推荐场景下原始数据集的特征归类需要。

4.1.1　数值型

数值型是最常见的特征类型，用户收入、商品评分、物品排名等特征通常都属于该类型。数值型特征的基本要求是特征值本身可以表示量的多少，并且在取值范围内存在加法或减法闭包算子。如果某特征满足该要求，即可将该特征标记为数值型特征。系统内置的数值型特征包含 4 个子类，下面将对每个子类的适用特征场景和构建所需的额外参数进行说明。

整型　整型值类型适用于取值范围内值的形式均为整型，且语义与数学整数定义相近的特征，例如每月登录天数、销量等特征。整型类型默认取值范围 $[-\infty, +\infty]$，支持额外参数 *bound*，用于规定特征的取值上下界。

连续浮点数型　连续浮点数类型适用于取值范围内值的形式均为浮点数，且语义与数学实数定义相近的特征，如用户收入、商品价格等。连续浮点数型默认取值范围 $[-\infty, +\infty]$，支持额外参数 *bound* 规定特征的取值上下界，默认四舍五入小数点后三位，支持额外参数 *round* 规定舍入精度。

离散浮点数型 连续浮点数类型适用于取值范围内值的形式均为浮点数，语义与数学实数定义相近，但取值不连续的特征，如服务满意度（一般以 0.5 为间隔）等特征。默认取值范围 $[-\infty, +\infty]$，支持额外参数 *bound* 规定特征的取值上下界，默认四舍五入小数点后三位，支持额外参数 *round* 规定舍入精度，默认间距 0.1，支持额外参数 *gap* 规定间隔。尽管离散浮点数类型或连续浮点数类型通过设置舍入精度和间隔值就可以满足整型功能，系统仍提供整型的目的是与系统使用者认知习惯保持一致。

日期型 日期型适用于特征特征类型为日期形式（不管分隔符为何种），或是具有日期语义的特征（如"19970824"）。日期类型默认取值范围是 $[(1970, 1, 1, 0, 0, 0, 0, 0, 0), (2020, 12, 31, 23, 59, 59, 0, 0, 0)]$，支持额外参数 *bound* 规定特征的取值上下界，*gap* 规定日期更改的最小幅度。

表 4.1: 数值型统计器

变量	类型	说明
bound	特征适配	设定取值区间
round	特征适配	浮点数精度裁剪
gap	特征适配	离散浮点数间隔
count	数据集分析	数据集中特征出现次数
type	基础设置	特征类型
max	数据集分析	数据集特征最大值
min	数据集分析	数据集特征最小值
average	数据集分析	数据集特征平均值
segment	数据集分析	特征分段统计

数值型统计器特征说明如表4.1所示，为了适配所有数值型特征，因此需要保留数值边界属性 *bound*、浮点特征精度裁剪参数属性 *round* 和浮点变更间距属性 *gap*。除此之外，统计器还需要统计特征出现的总次数 *count*，记录特征类型 *type*，分析数据集并更新特征值的实际区间 max, min 以及特征平均值 *average* 等。用户还可以配置统计器分段区间 *segment* 对特征值区间频数进行分桶统计，配置变异阈值 *delta* 对特征调整的“微小性”进行控制。

4.1.2　字符型

本质上，所有特征值在数据集文件中都采用字符串形式进行保存，因此判断某一特征是否属于字符型主要根据特征在具体场景下的语义。字符型特征的基本要求是特征语义非数值型或布尔型，且特征值不包括列表或字典（JSON）等形式的复杂结构。一般来说，字符型特征都具有某种组成格式或组成内容上的

约束。系统内置的字符型特征包含 4 个子类，下面将对每个子类的适用特征场景和构建所需的额外参数。

具有待选集且集合元素可穷举的字符串 该类型特征一般集合内元素有限。通常这类集合是公开的，比如用户所在省份和城市，用户的大学名称等。目前系统不提供内置集合，但支持额外参数 *list*，由用户提供特征取值集合列表。

具有待选集且集合元素不可穷举或未知的字符串 该类型特征虽然具有固定的待选集合且集合内元素有限，但这类集合是推荐场景内部使用的非公开集合，如推荐商品编号等。这类特征格式可能不固定，即使格式固定，部分符合格式的值也没有意义（如不存在的商品）。系统除了支持额外参数 *list*，由用户提供一定存在的特征取值集合列表，还会随机将特征值转换为其他相似用例中的对应特征值，用例相似性通过对不同类型特征相似值加权求和得到。

具有固定格式的字符串 该类型特征需要具有明确的固定格式，且符合格式的值都有意义，比如身份证号，手机号码等。系统支持且必须要求提供额外参数 *format*，用于随机生成特征值，且该参数必须满足正则表达式规范。

具有固定字符集的字符串 该类型特征既没有固格式，也没有取值集合，且通常字符串长度不定。因此系统默认为支持变异为英文、中文字符集，支持额外参数 *charset* 指定额外字符集。

表 4.2: 字符型统计器

变量	类型	说明
list	特征适配	特征值可选列表
format	特征适配	特征值正则表达式
charset	特征适配	特征值字符集
count	数据集分析	数据集中特征出现次数
type	基础设置	特征类型
segment	数据集分析	特征分段统计
max_length	用户配置	特征值长度上限

字符型统计器如表4.2所示，为了适配所有字符型特征，因此需要保留固定可选特征值集合 *list*、特征字符串正则表达式格式 *format*、字符串字符集 *charset* 属性。与数值型统计器一致，字符型统计器保留属性总次数 *count*，特征类型 *type* 以及统计器分段区间 *segment*。除此之外，为了限制固定字符集类型特征的规模，提供额外属性 *max_length* 限制特征值的最大字符长度。

4.1.3　布尔型

布尔型特征的具体形式多样，可能是 “是否”，可能是”0 1”，也可能是 “True False” ，但这些形式的语义一致，即能够表示逻辑上的真与假。统计器配置如表4.3所示，为了能够实现布尔特征的变化，统计器一方面记录了真假总次数 *count_true*、*count_false*，特征类型 *type*，真假表示形式 *format*。

表 4.3: 布尔型统计器

变量	类型	说明
count_true	数据集分析	正特征出现次数
count_false	数据集分析	负特征出现次数
type	基础设置	特征类型
format	用户配置	正负特征表示形式

4.1.4　列表型

列表型特征等价于一系列同类型特征的集合，集合内元素可能是数值型、字符型、布尔型等单特征集合，也可能是是列表型、字典型等复杂特征的集合，常见的列表型特征包括物品点击序列集合、用户购买商品集合等。列表型特征一方面可以采用其集合内元素所属类型的对应变异方法，实现内部元素值调整，另一方面可以采用列表型特征所属变异方法，从集合层面进行元素值和元素个数调整，改变集合内元素值的分布。

表 4.4: 列表型统计器

变量	类型	说明
count	数据集分析	数据集中特征出现次数
base_type	基础设置	特征类型
segment	数据集分析	特征分段统计
fix_length	用户配置	列表长度固定
bound	用户配置	设定取值区间
distribution_local	数据集分析	单条记录特征值分布
distribution_total	数据集分析	数据集特征值分布

统计器配置如表4.4所示，除了保留记录特征总次数的属性 *count*，和由用户提供的内部特征类型 *base_type* 外。还支持额外参数。*fix_length* 规定了列表长度无法修改，适用于诸如 “30 天是否登录” 这类定长列表特征。*bound* 设置了列表长度的上下界。*distribution_local* 记录了每条数据中列表内元素的分布，*distribution_total* 记录了数据集中列表内部元素的整体分布情况。

4.1.5　字典型

字典型特征也是一种组合特征，但字典内部可以由不同类型的特征构成，并通过键值对的形式组织特征名和特征值。字典型特征与列表型特征一样可以调用内部特征的变异方法来进行元素值修改，但集合层面的调整要更为复杂。字典型特征可以减少内部特征信息，但可能造成对原始数据语义完整性破坏，字典型特征可以增加内部特征，但备选特征不能超出可选特征集合。

表 4.5: 字典型统计器

变量	类型	说明
count	数据集分析	数据集中特征出现次数
feature_types	基础设置	特征类型
fix_inside	用户配置	特征个数固定
probability	数据集分析	特征值分布

统计器配置如表4.5所示，为了解决上述问题，字典型统计器除了统计总次数 *count*，还需用户提供 *feature_types* 属性，用于指明所有可能出现的内部特征名和特征类型。除此之外，*fix_inside* 规定了只允许修改内部特征值，不能对内部结构进行调整。*probability* 记录了数据集中所有内部特征出现的频率。

4.2　变异方法设计

针对上述丰富的内置特征类型，系统提供了与特征类型相对应的内置变异方法。在特征统计器完成初始化和数据集分析后，变异方法统计器属性信息完成对应特征值调整。为了控制变异方法调整幅度，用户设置统计器时提供的变异概率将在数据扩增中影响变异方法是否将被调用，而诸如数值特征下的变异幅度等参数将在调用变异方法后对调整规模进行控制。由于内置布尔型变异方法直接对原值取反，因此下面不对其进行详细介绍。

4.2.1　数值型

内置数值型特征变异方法的核心思想是基于用户提供的变异概率，变异幅度阈值和统计器分析结果进行特征值调整，调整形式以数值加减为主。

$$new_{int} = int[max(min(new, bound_{up}), bound_{down})] \tag{4.1}$$

$$new = origin \pm random_s * origin * (avg/length) \tag{4.2}$$

$$random_s = uniform(delta/2, delta) \tag{4.3}$$

$$length = max - min \tag{4.4}$$

整型 对于整型类型，调整幅度由 $random_s$ 和 $length$ 控制。$random_s$ 由用户提供的变异幅度阈值 $delta$ 控制，在 $delta/2$ 到 $delta$ 内以均匀采样方式生成随机数，$length$ 则为统计器遍历数据集得到的特征值实际取值范围，avg 为统计器遍历数据集计算得到的特征平均值，最后需要根据用户规定的上下界对特征值进行限制，并采用 $int[]$ 算子将最终结果转化为整型。

连续浮点数型 基于公式4.2，连续浮点数型需要对调整结果进行精度裁剪，得到公式4.5。其中 $[]_{round}$ 算子对浮点数进行四舍五入，保留小数点后 $round$ 位。

$$new_{round} = [new]_{round} \tag{4.5}$$

离散浮点数型 如公式4.6，$length$ 由公式4.4计算得到。离散浮点数型需要对 $delta$ 参数基于 gap 进行转化。其中 $< n >_{gap}$ 算子表示将浮点数 n 调整为 gap 的整数倍。

$$new = max(min(new_{tmp}, bound_{up}), bound_{down}) \tag{4.6}$$

$$new_{tmp} = origin \pm random_s * origin * (avg/length) \tag{4.7}$$

$$random_s =< uniform(delta_{gap}/2, delta_{gap}) >_{gap} \tag{4.8}$$

$$delta_{gap} =< delta >_{gap} \tag{4.9}$$

日期型 如公式4.10所示，日期型特征值的调整规模基于 gap 参数，gap 参数规定了特征值改变的最小粒度。

$$new_{date} = max(min(new, bound_{up}), bound_{down}) \tag{4.10}$$

$$new = origin \pm random_s \tag{4.11}$$

$$random_s =< uniform(delta_{gap}/2, delta_{gap}) >_{gap} \tag{4.12}$$

$$delta_{gap} =< delta >_{gap} \tag{4.13}$$

4.2.2 字符型

内置字符型特征变异方法的核心思想是基于用户提供的变异概率、相似用例中的对应特征值和统计器分析结果完成特征值调整，特征值的调整形式以字

符串中字符变化和字符串规模变化为主。

具有待选集且集合元素可穷举的字符串 如公式4.14所示，随机选择并将当前特征值调整为用户提供特征取值列表 *list* 中任意一个元素，采用的抽样方式为均匀随机抽样。

$$P(new = new_i) = \frac{1}{|list|},\ new_i \in list \tag{4.14}$$

具有待选集且集合元素不可穷举或未知的字符串 如公式4.15所示，对于数据集 *dataset* 中该文件下任意用例 *i*, 如果当前特征值 *origin* 与其相似性计算结果低于设定阈值 $max_{distance}$，则认为二者相似性高，可作为备选修改值。相似性计算公式如4.16所示，通过对一条用例中所有特征值的距离加权求和，得到数据集中任意两条用例间的相似性。相似性计算由统计器完成。

$$new = new_i,\ new_i \in \{\mathbf{i} | Similarity(\mathbf{i}, \mathbf{origin}) < max_{distance},\ i \in dataset\} \tag{4.15}$$

$$Similarity(\mathbf{i}, \mathbf{origin}) = \sum_{f \in Feature} w_i \cdot Distance(i_f, origin_f) \tag{4.16}$$

具有固定格式的字符串 针对该类型特征，内置变异方法直接依据正则表达式 $format$，随机生成生成新的特征值。

具有固定字符集的字符串 针对该类型特征，内置变异方法在用户提供的字符集中随机选择任意个字符，以此在原始字符串的任意位置进行随机数量的增删字符或修改字符。

4.2.3　列表型

列表型特征提供的变异方法能够随机选择内部元素，并基于列表型统计器属性 *base_type* 指定的内部特征类型，采用相应的变异方法对单个元素进行调整。除此之外，如果用户设定 fix_length 参数允许列表长度变更，变异方法还能够利用统计器得到的特征值分布数据对列表内部元素进行复制扩增或删除。

4.2.4　字典型

字典型特征的变异方法一方面基于参数 $feature_types$ 指定的内部特征类型，一样采用内部特征的变异方法对单个元素进行调整，另一方面利用统计器 fix_inside 和 *probability* 调整内部参数的个数。由于字典型可能存在嵌套结构，变异方法采用逐层分析，并只对最内层特征个数进行调整。

4.2.5　类型无关方法

类型无关方法从整个数据集的用例分布出发，着重关注正负样本在数据集中的比重。系统内置的相应变异方法为正负均衡和正样本增强，用户使用前需要提供额外参数 $label_{col}$，为系统指定标签列位置。

正负均衡 由于数据集中正负样本比重通常不均衡，可能导致评估结果质量具有偏向性。为了解决这一问题，系统内置的正负均衡方法采用分层抽样法对负样本信息量进行重要性划分，每层以不同比重进行抽样，最终使得数据集中正负样本数量级一致。

正样本增强 该方法直接从正样本中随机抽样，并采用特征结构中定义的特征类型所属变异方法进行微小扰动，参数 k 指定扩增倍数。

4.3　评估指标设计

为了在推荐模型模糊测试完成后，向用户呈现更全面、更深入的模型离线执行评估报告，系统内置了如下评估指标。指标的展现形式包括量化值和图表，并且除了通用指标外，系统还内置了多个特色指标。

4.3.1　准确率

准确率是最常用的评估指标，将任意物品对用户进行推荐，用户对物品点击或购买行为的预估概率都可以通过阈值（一般为 0.5，也可进行上调或下调）划分为正样本或负样本，从而转化为一个二分类问题。准确率计算公式如4.17所示，$n_{correct}$ 表示正确分类的样本个数，n_{total} 为总样本数。

$$Accuracy = \frac{n_{correct}}{n_{total}} \tag{4.17}$$

4.3.2　精准率和召回率

精准率（Precision) 表示用例正样本判定正确的概率，召回率（Recall）则表示成功判断的正样本占数据集中所有正样本的比重。由于推荐模型执行结果通常不是一个物品输出，而是以 TopN 的形式输出多个物品，因此 Precision-N 和 Recall-N 指标更有实际价值（一般会评估多个 N 情况，如 5/10/30，根据业务场景需要而定)。Precision-N 计算公式如4.18所示，Recall-N 如4.19所示，其中 TP_N 表示 TopN 真正用例数（判断为真实际也为真)，FP_N 表示 TopN 假正用例数（即

判断为真实际为假），FN_N 表示 TopN 假反用例数（即判断为假实际为真）。

$$Precision = \frac{TP_N}{TP_N + FP_N} \tag{4.18}$$

$$Recall = \frac{TP_N}{TP_N + FN_N} \tag{4.19}$$

除此之外，由于精准率和召回率内部的矛盾性，业界还常用 F1 指标基于二者进行综合衡量，公式如4.20所示。

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.20}$$

同时，基于 Precison-N 和 Recall-N，系统提供 P-R 曲线绘制，每个离散点都为不同阈值下计算结果，同时给出 AUC 数据进行量化。

4.3.3　平均精度均值

精度均值（AP）表示推荐模型为某一用户生成推荐列表下，推荐精准率的平均值，计算公式如4.21所示，其中 $Precision_i$ 表示某一具体用户推荐列表下 $Topi$ 推荐精准率。精度均值相比于 Precision，考虑了用户推荐列表中的物品顺序，反映了推荐模型针对具体用户的精排质量。而平均精度均值（MAP）则对所有用户的平均精度取平均值，从而反映推荐模型的整体排序质量。平均精度均值计算公式如4.22所示，其中 m 表示用户总数，AP_i 表示用户 i 的推荐精度均值。

$$AP = \frac{1}{N} \sum_{i=1}^{N} Precision_i \tag{4.21}$$

$$MAP = \frac{1}{m} \sum_{i=1}^{m} AP_i \tag{4.22}$$

4.3.4　热度值与热度分桶

热度，即推荐模型中被推荐给 N 个以上用户的物品个数占用户推荐物品总次数的比例。热度计算公式如4.23所示，其中 n_{hot} 表示达到热度阈值 hot 的物品个数，N 表示推荐列表长度，U 表示用户个数（抽样可以降低运算规模）。通常来说，为了减少长尾比重，提高推荐多样性，热度值应当较低。同时，为了直观反映推荐热度分布，我们将物品根据推荐次数进行分桶，分桶数由用户指定。

$$Hot = \frac{n_{hot}}{N \cdot U} \tag{4.23}$$

4.3.5　t-SNE 降维

推荐模型通过特征工程将原始特征转化为 Embedding 作为模型输入，而 t-SNE 作为一种非线性降维算法，将高维 Embedding 映射为二或三维空间点，高维 Embedding 的相似性通过降维空间点的距离体现。因此 t-SNE 结果能够反映特征工程质量，具有可解释性。t-SNE 首先针对高维输入对构建分布概率模型，概率值与相似性成正比，然后在低维空间构建相似分布模型，最后生成散点。系统内置 t-SNE 计算采用 sklearn.manifold.TSNE 方法实现。

4.4　本章小结

本章节介绍了系统内置的特征类型，说明了内置类型的划分依据和适用范围。除此之外，还说明了开发者使用内置特征类型必须或可以提供的额外参数，以及特征统计器如何完成原始数据集中的特征信息分析。本章节介绍了系统内置的变异方法，说明了变异方法的变异原理和用户提供参数的影响。本章节介绍了内置评估指标计算原理和表现形式。

第五章　系统详细设计与实现

5.1　数据流模块设计与实现

5.1.1　流程设计

数据流模块包含两个子部分：文件上传和文件下载。上传部分支持用户提交测试文件，在对文件进行合法性检查后，完成文件的服务器本地化存储和持久化数据更新。下载部分主要响应用户的数据下载请求，为用户打包并传输模糊测试执行过程中生成的，存储在服务器本地的扩增测试数据集。

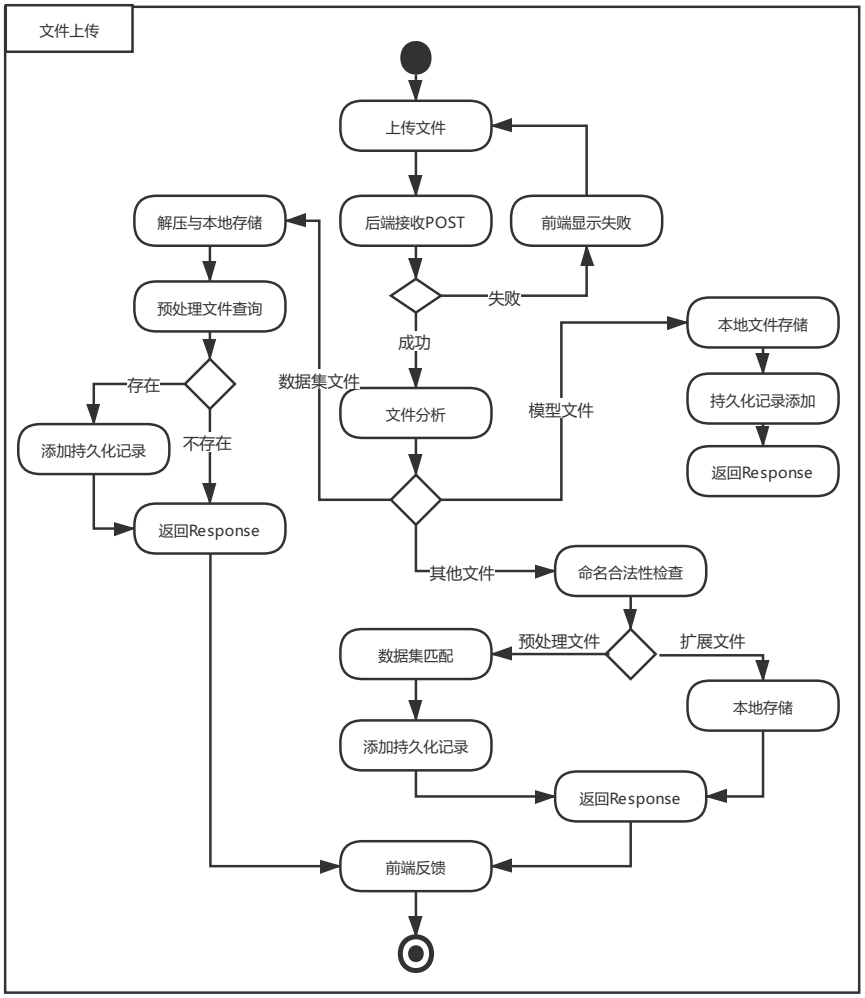

图 5.1: 文件上传流程图

图5.1为文件上传流程图。用户通过拖拽文件，或浏览添加本地文件的方式选择上传文件，后端成功接收到前端请求后，针对文件类型和用途进行详细分析。如果用户上传了数据集文件，系统首先完成解压与服务器本地存储。接着查询本地文件，寻找与之匹配的数据集预处理文件。如果存在，则添加数据集持久化记录，此时该数据集可被用于构建测试任务。如果用户上传了模型文件，系统直接储存到服务器本地指定位置，并新增模型持久化数据，完成后该模型即可在后续测试任务构建中作为测试对象。如果用户上传了其他文件，需要进行文件名合法性检查，以分析其名称是否符合数据集预处理文件或扩展文件的命名规范。如果符合预处理文件命名规范，则保存到服务器本地对应位置，并与本地已有但未完成匹配的数据集进行匹配，如果匹配成功则添加数据集持久化记录。如果符合功能扩展文件，则根据其属于特征类型划分或变异方法存储到服务器相应位置。最后服务器向前端发送包含执行结构信息的 Response，前端接收后进行解析，并向用户提供相应反馈。

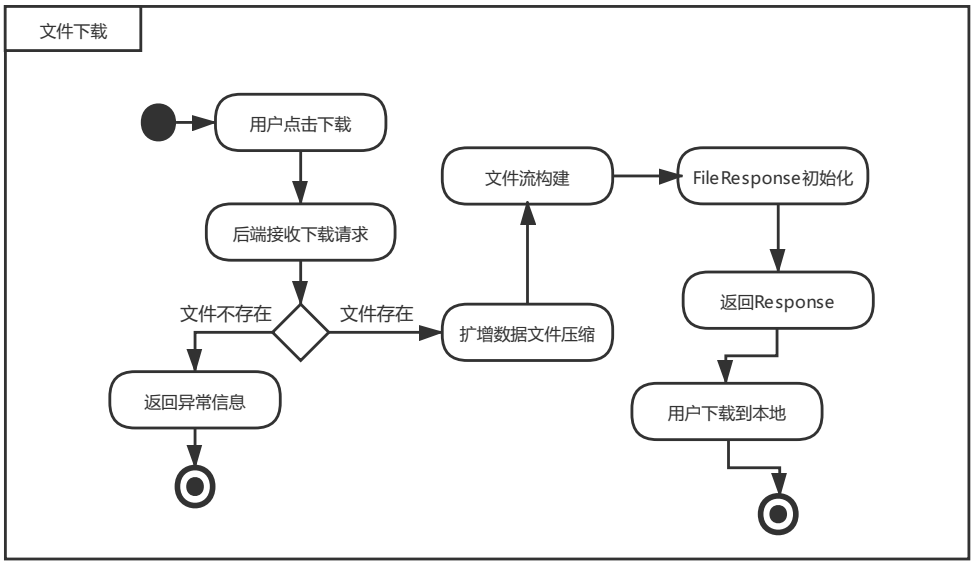

图 5.2: 文件下载流程图

图5.2为文件下载流程图。用户点击前端页面下载列表中的“下载”按钮后，前端向服务器发送下载请求。服务器接收请求后首先进行文件存在性检查，如果文件存在则生成压缩包，并构建文件流对象，然后生成 FileResponse 对象并返回给前端，前端用户将文件下载到本地指定位置。

5.1.2　时序图

数据流模块中文件上传流程时序图5.3所示。在后端成功接收到用户的 upload 请求后，FileUploadService 首先完成文件类型分析，再调用 FileCheckClient 的 check_file(file_name) 完成文件命名合法性验证。如果文件属于数据集或模型类型，则 check_file(file_name) 直接返回 True，如果是预处理文件或扩展文件，将检查文件命名是否符合相应规范。如果文件通过检查，则 FileUploadService 调用 FileCheckClient 的 file_analysis(file) 方法处理上传文件。无论用户提交的文件是哪种类型，首先调用 LocalSaveService 的 save() 方法完成存储路径分析，从而将文件存储到服务器本地指定位置。接着针对数据集文件和数据预处理文件，file_analysis() 方法完成匹配工作，如果匹配通过则通过 ORM 新增持久化记录。最后将执行结果生成 JSONResponse 对象返回给发送方。

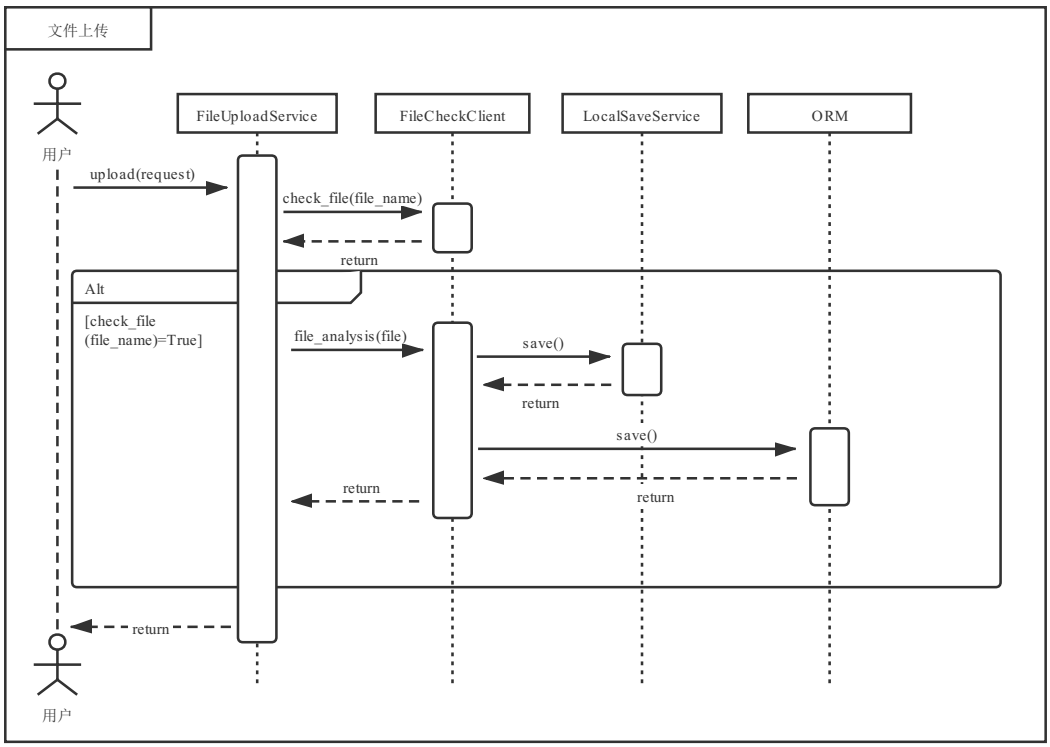

图 5.3: 文件上传时序图

数据流模块中文件下载流程的时序图如图5.4所示。当用户进入”下载扩增数据“页面并点击页面中的”下载“按钮后，前端将会调用 DownloadService 中的 download() 方法。DownloadService 首先根据 Request 提供的 GET 请求参数 task_id，调用 FileCheckClient 的 get_task_info() 方法查看任务记录是否存在，get_task_info() 方法将调用 ORM 的 filter() 方法，采用 task_id 筛选并返

回持久化记录。如果任务记录存在，则根据任务记录中的相应字段，调用 LocalSaveService 所提供的 get_download_info() 方法，查询对应的数据库下载表信息，get_download_info() 方法调用 ORMfilter() 方法获取下载记录并返回。如果下载记录存在，那么 DowloadService 将下载路径作为参数，调用 LocalSaveService 的 download() 方法，此时 LocalSaveService 将构建并返回文件流对象。最后 DownloadService 向发送方返回 FileResponse，包含下载文件或异常信息。

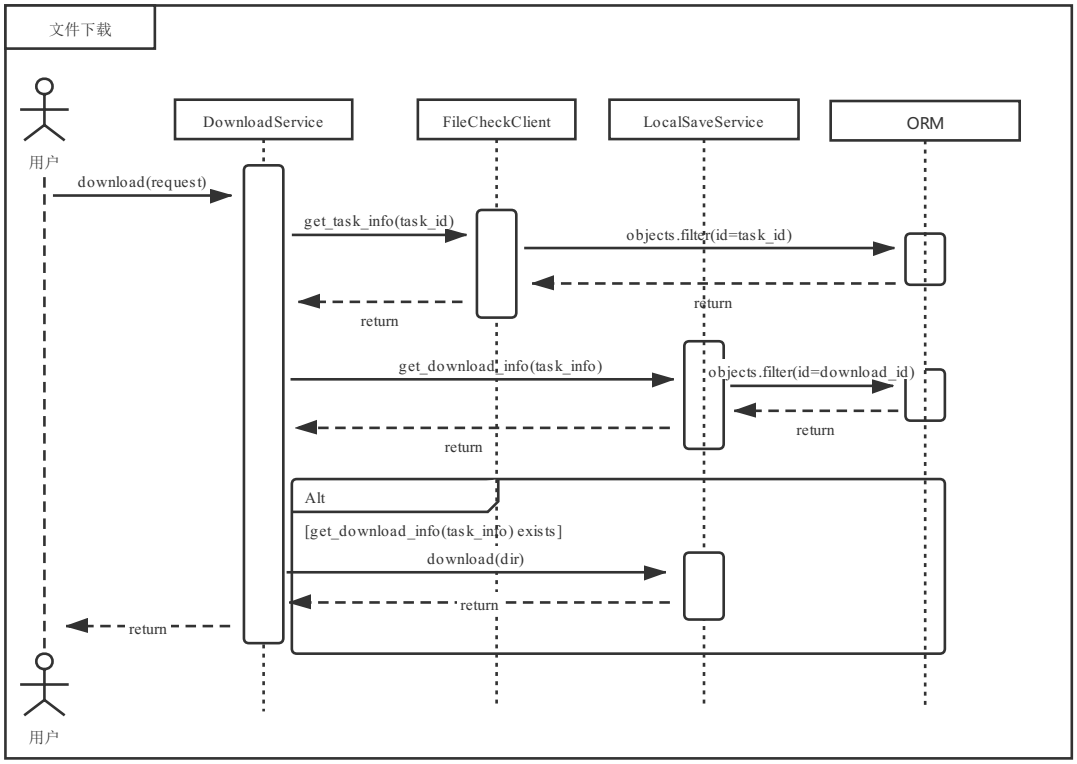

图 5.4: 文件下载时序图

5.1.3　关键代码

上传关键代码如图5.5所示，其中包含了两个关键实现方法，file_unzip() 负责对上传文件解压并存储到服务器本地，file_analysis() 对用户上传文件的命名合法性进行检查并新建持久化记录。

file_unzip() 方法首先采用 Python zipfile 包中自带的类型检查方法 is_zipfile()，确认文件是 zip 格式，并采用 zipfile 中的 ZipFile 对象打开文件。打开后为文件创建解压结果存储文件夹，并遍历压缩文件中每个文件。对于其中任意一个文件或文件夹，获取其大小以区分其类型是文件或文件夹，然后采用 gbk 编码解决中文转码问题，最后拼接解压文件路径并进行存储。

file_analysis() 给出了对于预处理文件分析的具体实现。首先需要对文件名进行分析处理，通过 split('.') 方法去除掉类型后缀，并采用 split('_') 进行分隔，如果命名以" reader "结尾，则系统判定命名合法。接着获取存储在" dataset "目录下的所有数据集文件夹，并分析数据集名称与预处理文件名称是否匹配。如果匹配，则构建 ORM 模型对象 Dataset，并调用 save() 方法添加相应的持久化记录。

```
def file_unzip():
    # 如果是 zip 则解压
    is_zip = zipfile.is_zipfile(save_path)
    if is_zip:
        with zipfile.ZipFile(file=save_path, mode='r') as zf:
            #创建解压目录
            os.mkdir(get_folder_name(save_path) + '/datasets/' + get_file_name(save_path))
            for old_name in zf.namelist():
                # 获取文件大小
                file_size = zf.getinfo(old_name).file_size
                # 由于源码遇到中文是 cp437 方式，解码成 gbk
                new_name = old_name.encode('cp437').decode('gbk')
                # 拼接文件的保存路径
                new_path = os.path.join(get_folder_name(save_path)
                                        + '/datasets/'
                                        + get_file_name(save_path), new_name)
                # 判断文件是文件夹还是文件
                if file_size > 0:
                    with open(file=new_path, mode='wb') as f:
                        f.write(zf.read(old_name))
                else:
                    os.mkdir(new_path)
def file_analysis():
    if ('.'.join(file.name.strip().split('.')[:-1])).split('_')[-1] == 'reader':
        dataset_list = os.listdir(get_folder_name(save_path) + '/datasets/')
        upload_dataset_name = ''.join(file.name.strip().split('_')[:-1])
        for dataset in dataset_list:
            if dataset.lower() == upload_dataset_name:
                new_dataset = DataSet(dataset_name=dataset)
                new_dataset.save()
```

图 5.5: 上传流程关键代码

下载关键代码如图5.6所示。download() 方法首先从 Request 对象中拆解出 task_id 参数，并根据 task_id 参数调用 get_task_info() 方法获取 ORM 任务模型实例 task。接着 get_download_info() 方法利用 task 实例的 result 属性值，获取 ORM 下载模型实例 file，并返回 file 实例的本地路径 file_dir 属性。最后 download() 方法构建文件流 FileResponse，配置 header 的”Content-Type“和”Content-Disposition“属性，并将 FileResponse 对象返回给前端。

```
def download(request):
    if 'task_id' in request.GET and request.GET['task_id']:
        task_info = get_task_info(request.GET['task_id'])
        file_dir = get_download_info(task_info)
        file_stream = StreamingHttpResponse(open(file_dir, 'rb'))
        response = FileResponse(file_stream)
        response['Content-Type'] = 'application/octet-stream'
        response['Content-Disposition'] = 'attachment;filename="{0}"'.format(file_dir)
        return response
```

图 5.6: 下载流程关键代码

5.1.4　界面演示

图 5.7: 文件上传页面展示

如图5.7所示，用户点击侧边栏的”上传数据集与模型“选项进入文件上传页面，采用拖拽或”点击上传“提交文件，演示中上传了 AmazonMusic 数据集和对应的 amazon_music_reader.py 预处理文件，以及模型文件 model_test 和自定义类型划分 test_adding_recorder。右侧出现绿勾表示文件上传成功。

如图5.8所示，用户点击侧边栏的”下载扩增数据“选项进入扩增数据集下载页面，页面给出了所有可下载的任务列表。通过左侧的复选按钮，点击”批量删除“可以完成列表中勾选记录的快速清理。用户可使用状态筛选器和任务名搜索框相结合，完成模板任务记录的快速定位。针对此处给出的每条测试任务记录，都可以点击”下载“按钮将数据集下载到本地，如图中所示的 fake_data_test_amazon_music.zip。除此之外，还可以点击”删除“按钮，进行已完成测试任务和相应扩增数据的逐条删除。

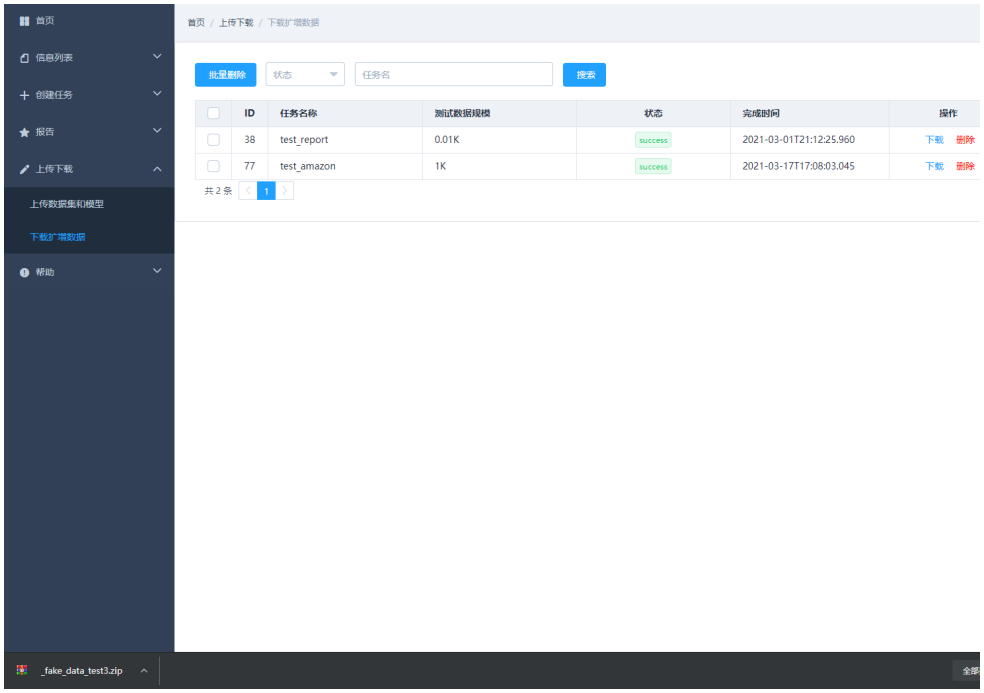

图 5.8: 文件下载页面展示

5.2　模糊测试任务模块设计与实现

5.2.1　流程图

特征结构构建流程如图5.9所示。当用户在浏览器完成特征结构构建，并点击提交按钮提交结构信息后，服务器首先解析请求数据对象 Request 内部结构，检

查 Request 的信息完整性，如果不完整，服务器将返回包含异常信息的 Response，并由前端向用户给出错误提示反馈。完整的 Request 将被系统抽取出 body 部分并进行解析，从而为每个特征单体新建 ORM 模型对象，并以此新增持久化特征记录。接着系统解析 body 中包含的特征关联性信息，将特征单体的关联关系映射为特征记录间的关联关系，并通过相应的特征单体模型对象更新特征持久化记录信息。最后服务器向前端返回包含执行结果的 Response。

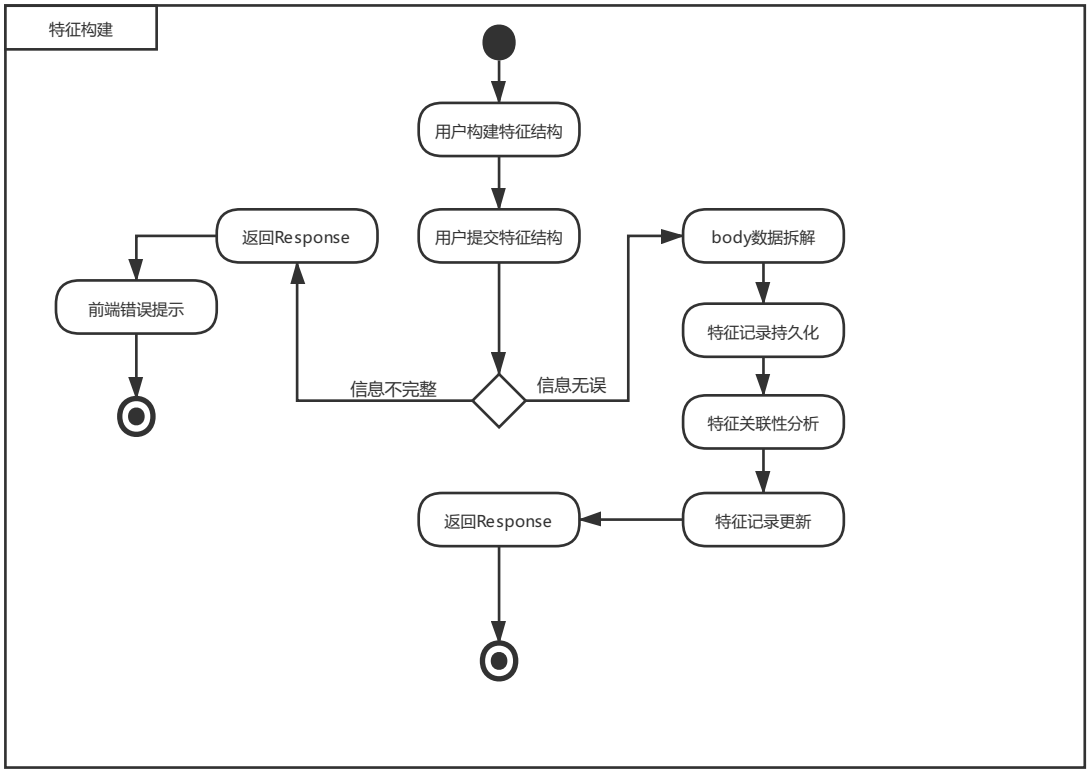

图 5.9: 特征构建流程图

任务创建流程图如图5.10所示。服务器接收到测试任务提交请求并验证 Request 信息完整性后，需要进行 Request 对象的 body 拆解，提取出模型、数据集、特征结构、测试规模等相关参数。系统首先根据数据集名称和持久化信息，动态调用数据集预处理文件，执行数据集内容校验，并去除内部不合法的字符。然后系统生成测试任务对象，并新增相应的持久化任务记录。接着，系统利用持久化特征结构信息，构建测试任务所需的特征对象，并根据用户提供的特征参数初始化特征统计器，同时遍历数据集中的相应文件完成特征计数器的属性更新。当完成特征对象构建后，如果使用者设定任务为关联测试模式，系统还需根据特征关联关系，构建特征树对象，用于变异过程的特征值协同调整。最后，系统开始执行模糊测试任务。

模糊测试流程包含如下几步。首先，系统从原始数据集中随机抽取测试用例，并采用用户设定的特征变异方法进行属性扰动，生成新的测试数据。新测试数据被添加到原始数据集中，得到扩增测试数据集。然后，系统将扩增测试数据集作为待测模型的输入并执行。接着，系统基于模型执行结果，计算推荐模型质量评估指标，并将量化值和散点信息转化为持久化记录并存储。最后，系统更新测试任务持久化记录的完成时间、执行状态和执行结果字段，并返回 Response。

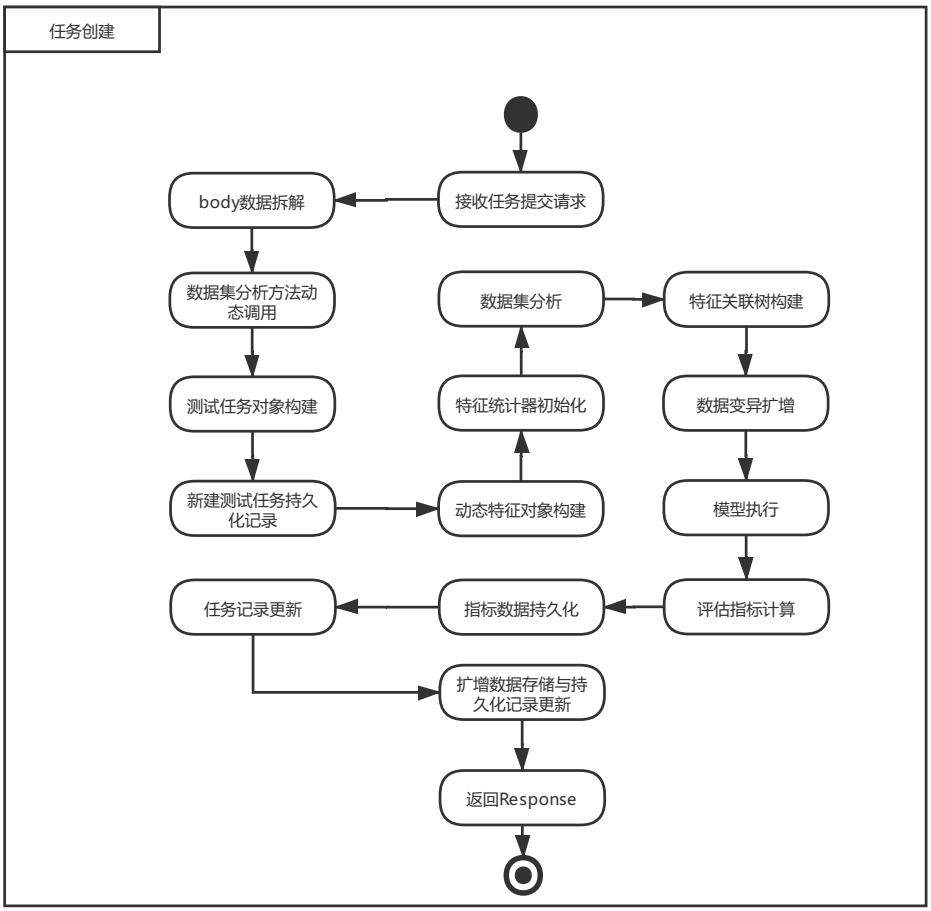

图 5.10: 任务创建流程图

5.2.2　时序图

特征结构创建流程的时序图如5.11所示。用户调用 FeatureStructureService 下的 add_structure() 方法新增特征结构。首先，FeatureStructureService 调用 StructureClient 的 create() 方法，抽取出 body 信息中的特征单体信息，利用这些信息构建特征模型的 ORM 实例，并调用 ORM 的 save() 方法新建持久化特征数据。接着，FeatureStructureService 调用 StructureClient 的 relation_analysis() 方法，抽

取出 body 信息中的特征关联信息，并将 body 中基于特征单体 ID 的关联关系映射为基于特征单体持久化记录 ID 的关联关系，以此对持久化记录的相关字段进行更新。最后，FeatureStructureService 向发送方返回 Response。

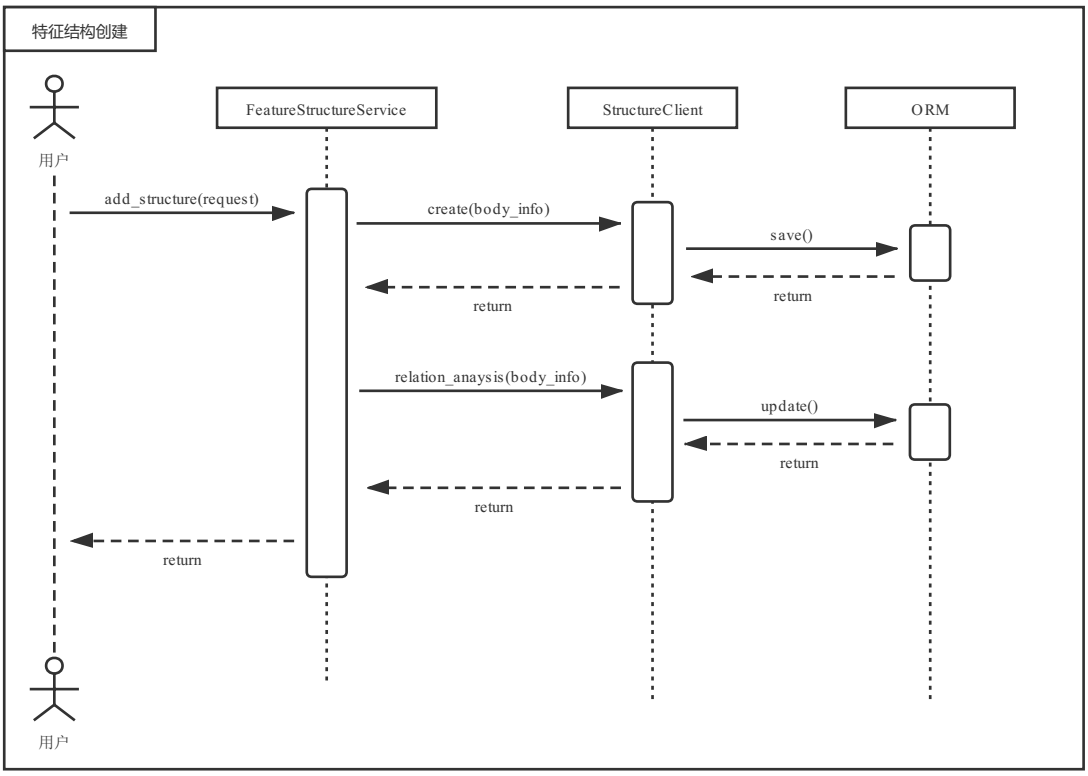

图 5.11: 特征构建时序图

任务构建流程时序图5.12所示。用户通过浏览器调用 TestTaskService 的 add_task() 方法，提交新建测试任务请求。首先，TestTaskService 调用 dataset_analysis() 方法完成数据集预处理，调用 create_task() 方法完成测试任务对象创建和初始化，并调用 ORM 的 save() 方法新增测试任务持久化记录。接着，TestTaskService 调用 create_features() 方法构建特征对象，调用 create_recorders() 完成特征对象的统计器初始化和基于数据集分析的统计器属性更新。然后，系统开始执行模糊测试流程。TestTaskService 调用 DataMutateService 的 mutate() 方法，多次抽取数据集中的测试用例，并利用相应的特征变异方法进行用例调整，得到扩增测试用例，直到扩增用例规模达到用户期望。此时 DataMutateService 调用 LocalSaveService 的 save() 方法，将扩增测试用例保存到服务器本地。完成数据扩增后，TestTaskService 调用 ModelRunService 的 run() 方法，采用扩增测试数据作为模型输入，此时 ModelRunService 需要通过 ORM 的 filter() 方法获取模型信息，并通过 LocalSaveService 的 load_model() 方法加载本地模型，最终完成模型执行

并收集执行结果信息。结果信息将被 TestTaskService 通过 evaluate() 方法传递给 ReportService，转化为指标结果，并通过 ORM 的 save() 方法进行持久化存储。最后 TestTaskService 调用 ORM 的 update() 方法更新任务记录的完成时间、任务状态和任务结果字段，并调用 ORM 的 save() 方法，将扩增数据集在服务器本地的存储路径，转化为数据库中的持久化记录。

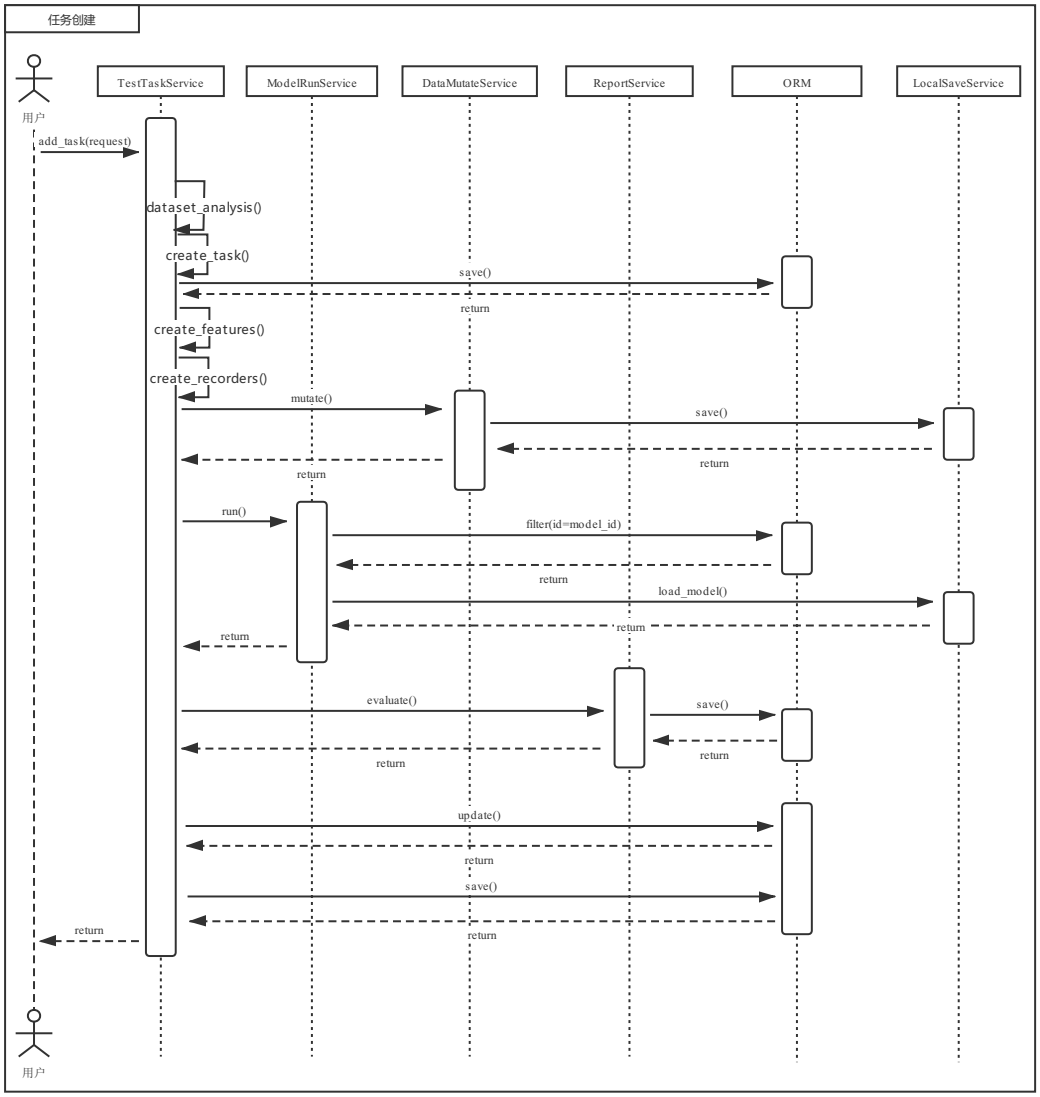

图 5.12: 任务创建时序图

5.2.3　关键代码

图5.13给出了构建特征结构过程的关键代码。create() 方法负责新增持久化记录。通过解析 body 中 scene 属性下的 blocks 子属性，获取特征单体信息，并

从特征单体信息中解析出特征参数部分。然后利用特征参数创建特征结构模型 Structure 的实例，并设置关联列表 related_ids 为空。最后调用 save() 方法新增持久化记录。relation_analysis() 方法负责特征关联性解析，对于包含于 scene 属性下的 blocks 子属性中的关联信息，get_idx_by_origin_id() 方法将原始特征块的 ID 值转化为持久化特征记录的 ID 值。接着根据解析结果更新特征实例的 related_ids 信息。最后调用 save() 方法完成更新，等价于时序图中 update() 语义。

```
def create(body):
    form_data = json.loads(body)
    # 拆解 body
    for scene in form_data['scene']['blocks']:
        scene = scene['values']['property']
        tmp = Structure(user=user_name,
                        structure_name=structure_name,
                        file_name=scene.get('file_name').get('value'),
                        feature_name=scene.get('name').get('value'),
                        feature_type=scene.get('feature_type').get('value'),
                        adversarial_args=scene.get('adversarial_args').get('value'),
                        adversarial_func=scene.get('adversarial_func').get('value'))
        tmp.set_related_ids([])
        tmp.save()
def relation_analysis(body):
    # 解析 related 部分
    for link in form_data['scene']['links']:
        start_idx = get_idx_by_origin_id(form_data['scene']['blocks'], link['originID'])
        end_idx = get_idx_by_origin_id(form_data['scene']['blocks'], link['targetID'])
        if scene_list[end_idx].related_ids == '[]':
            scene_list[end_idx].related_ids = []
        scene_list[end_idx].related_ids.append(scene_list[start_idx].id)
    for record in scene_list:
        if record.related_ids != '[]':
            record.set_related_ids(record.related_ids)
            record.save()
```

图 5.13: 构建特征结构关键代码

5.2.4　界面演示

特征创建页面如图5.14所示。用户首先需要提供特征结构名称，接着通过右击新建特征块，并在右侧红框中填写相应参数。填写完成后点击 save 按钮进行

保存。特征块可以通过点击右上角的红叉进行删除，当需要构建特征关联时，只需将关联特征的 in 和 out 端完成连接即可，连线方向是逻辑依赖关系主次顺序。当特征结构构建完成后，点击“立即创建”按钮进行提交。

任务创建页面如图5.15所示，用户提供任务名称，依次选择测试模型、数据集、特征结构和具体特征，并填写测试规模参数和选择测试类型后，即可点击“立即创建”按钮提交测试任务。

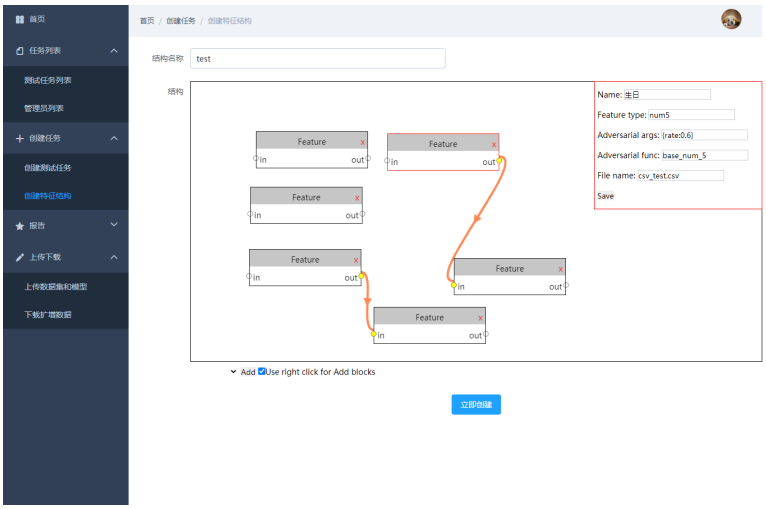

图 5.14: 特征创建页面展示

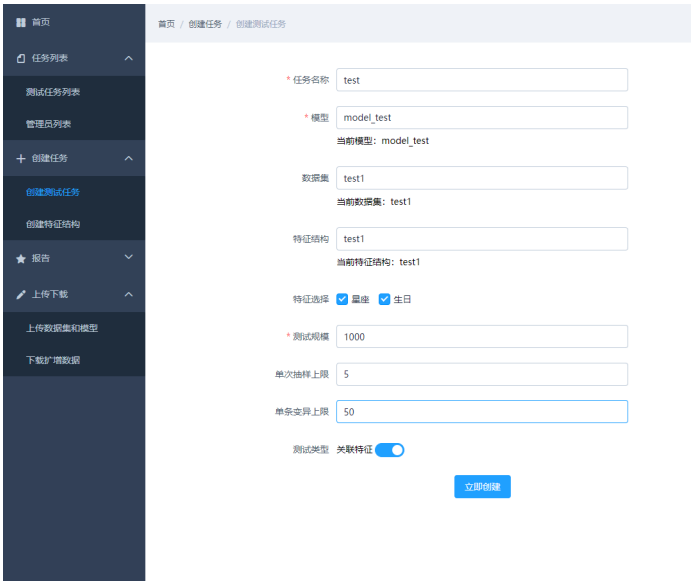

图 5.15: 任务创建页面展示

5.3　任务管理模块设计与实现

5.3.1　流程图

任务管理流程图如图5.16所示。系统响应用户在任务管理页面下的具体操作，如果用户使用筛选器进行基于任务执行状态的任务列表筛选，或是采用搜索框搜索任务名称，前端页面都会收到由服务器返回的 Response，其中包含最新用户所属的全部测试任务信息。此时交由前端完成数据检查和数据筛选，并完成展示测试任务列表刷新。如果用户选择查看任务详情，则服务器同样返回包含任务详细信息的 Response。前端负责完成对 Response 中解析出任务详情信息，完成详情页面的渲染和页面跳转。

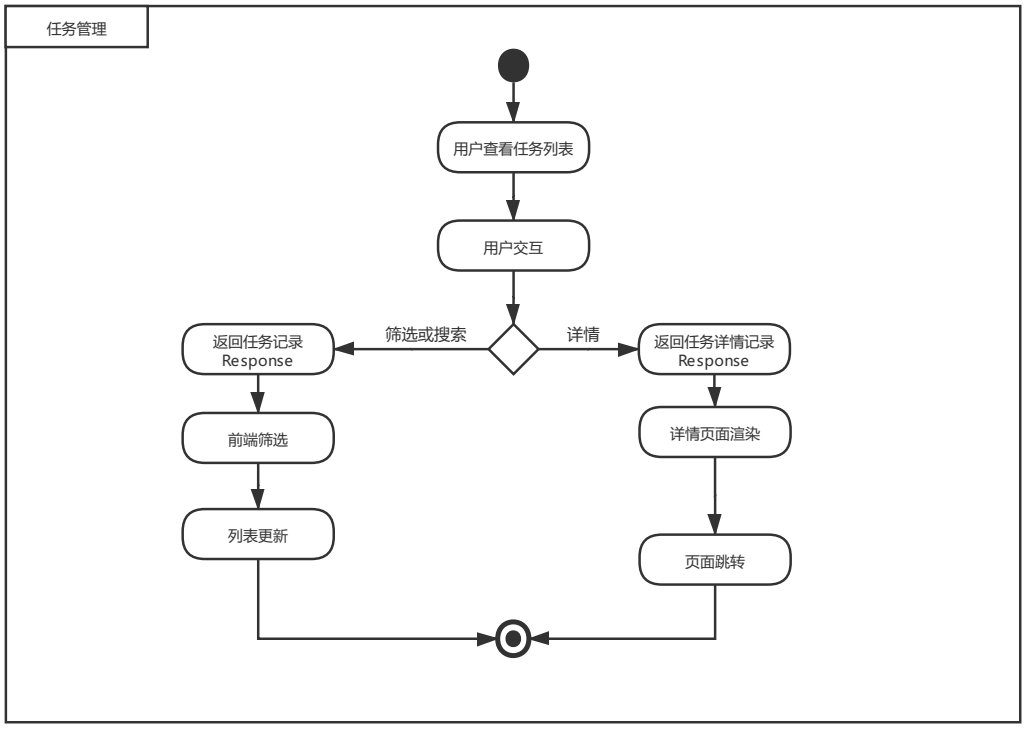

图 5.16: 任务管理流程图

5.3.2　时序图

任务管理时序图如5.17所示。当用户进入任务管理页面，或是用过滤器和搜索框进行任务筛选时，调用 ShowTaskService 的 get_task_info() 方法，通过 ORM 获取用户所属全部任务数据。如果用户点击详情，则通过 get_detail() 方法获取

单条任务详情信息。除此之外，当用户点击报告按钮后，调用 ShowTaskService 的 show_report() 方法，进行报告指标获取。

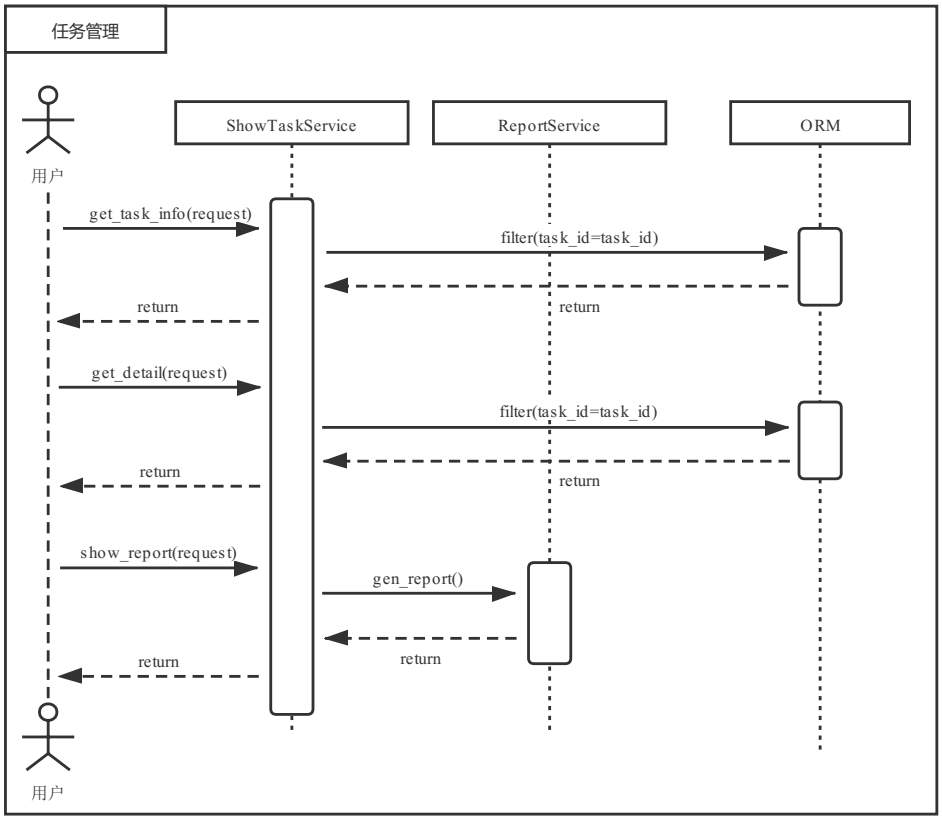

图 5.17: 任务管理时序图

5.3.3　界面演示

任务管理页面如图5.18所示。通过筛选并采用测试任务名称 “test_mutate” 进行精准定位后，得到了唯一匹配的测试任务记录。

5.4　报告生成模块设计与实现

5.4.1　流程图

报告生成流程图如5.19所示。用户无法直接通过侧边栏进入测试报告页面获取测试报告，而是首先进入任务管理页面查看任务列表，定位所需查看报告的具体任务记录，然后点击报告按钮。此时服务器响应前端请求，通过测试任务 ID 查找对应的评估指标记录，并利用评估指标记录 ID 获取所有绘图所需的散

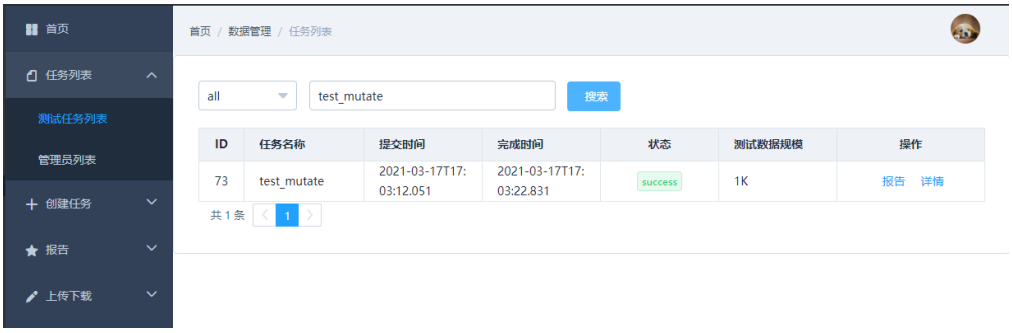

图 5.18: 任务管理页面展示

点记录。最后构建前端约定的字典结构，并作为 Response 参数返回给前端。前端页面解析 Response 完成页面跳转和渲染。

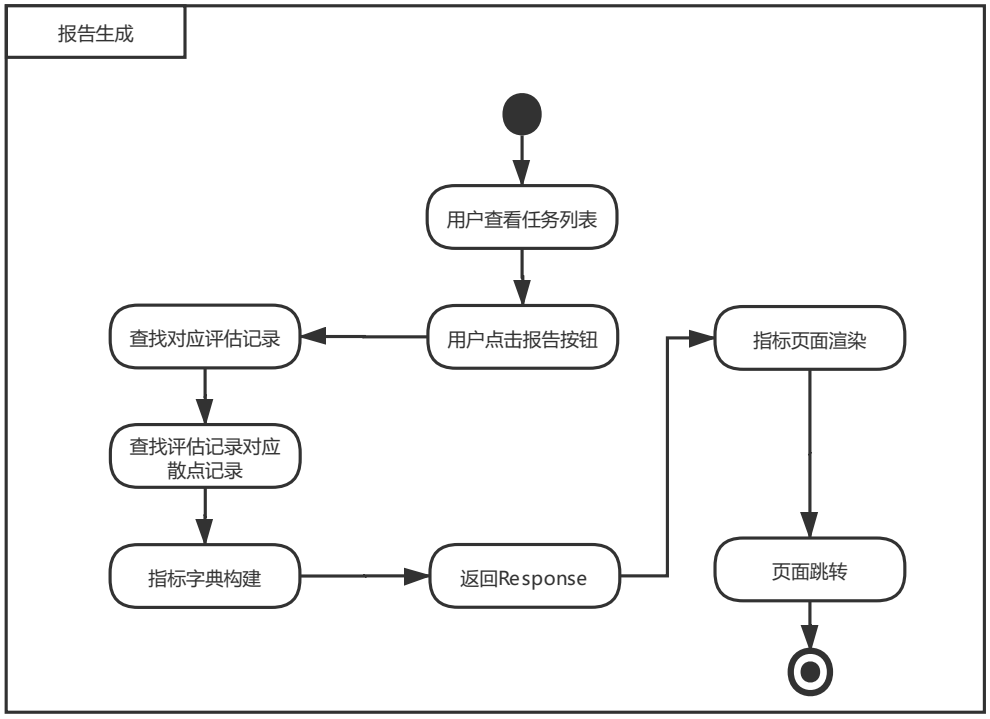

图 5.19: 报告生成流程图

5.4.2　时序图

报告生成时序图如5.20所示。用户点击报告按钮后，调用 ReportService 的 get_report() 方法获取持久化质量评估记录。在这一过程中，一方面 ReportService 调用 ORM 的 filter() 方法，根据测试任务记录中的评估结果字段值，获取包含量

化指标结果的评估记录信息，另一方面通过当前评估记录的 evaluation_id 调用 ORM 的 filter() 方法，过滤散点表获取对应的散点数据用户图表绘制。最后返回包含指标信息的 Response。

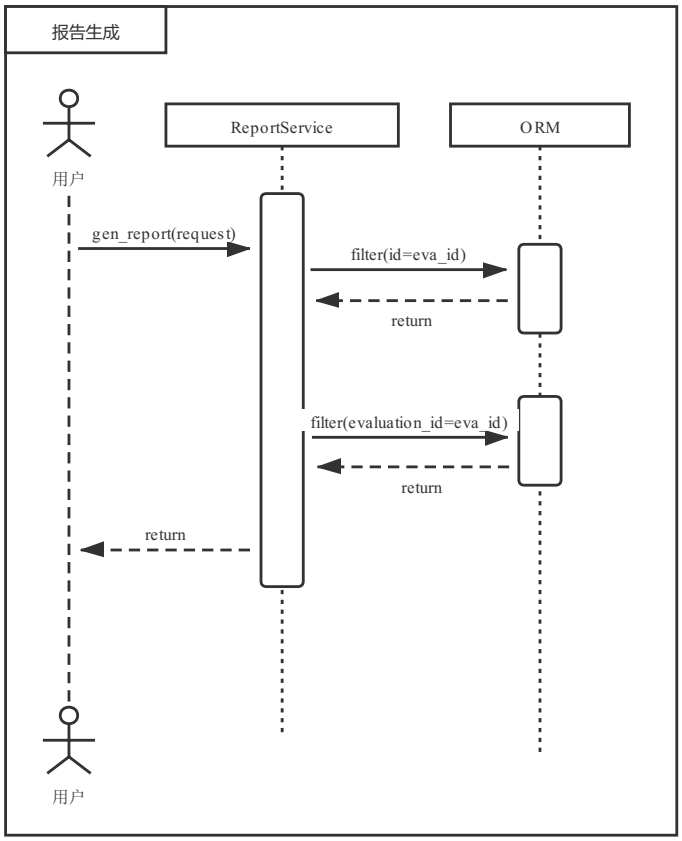

图 5.20: 报告生成时序图

5.4.3　关键代码

该部分关键代码如5.21所示，这段代码综合了两次 filter 操作。根据 task_id 获取 eva_info，基于 eva_info 调用散点表模型 EvaluationPoints 的 filter() 方法，获取折线图散点 lines 和散点图散点 scatters。

5.4.4　界面演示

报告生成页面如图5.22所示。系统自动化生成的测试报告向用户提供测试任务基本信息，包括用户名、任务名、耗时、测试规模和完成时间。除此之外，报告包含量化指标 AUC、精准率、召回率、F1、MAP 和热度值，提供 P-R 曲线和物品热度分桶，以及 t-SNE 可解释性分析结果。

```
def get_evaluation_info(request):
    task_id = request.GET['task_id']
    response = {'evaluation': {}, 'line_chart': [], 'scatter_chart': []}
    eva_info = Evaluation.objects.get(task_id=task_id)
    response['evaluation'] = model_to_dict(eva_info)

    lines = EvaluationPoints.objects.filter(evaluation_id=eva_info.id, figure="line")
    for line in lines:
        response['line_chart'].append(model_to_dict(line))

    scatters = EvaluationPoints.objects.filter(evaluation_id=eva_info.id, figure="scatter")
    for scatter in scatters:
        response['scatter_chart'].append(model_to_dict(scatter))
    return JsonResponse(response)
```

图 5.21: 报告生成关键代码

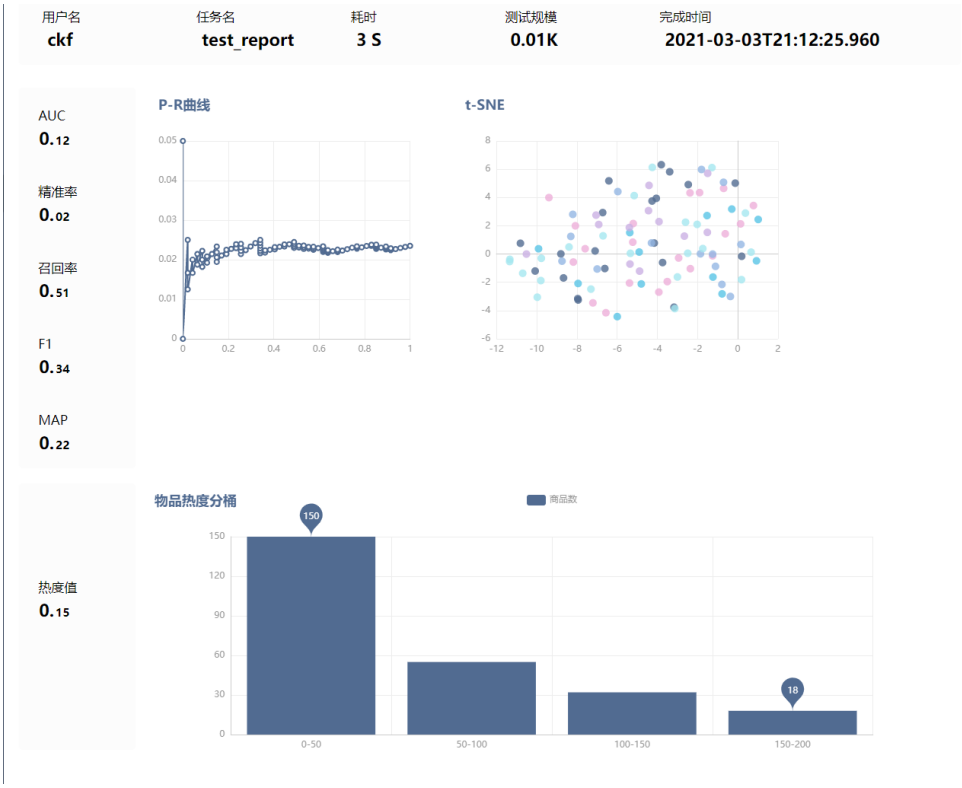

图 5.22: 报告生成页面展示

5.5　本章小结

本章节着重介绍了数据流处理模块，模糊测试任务模块，任务管理模块和报告生成模块的详细设计与实现。针对每个具体模块，通过流程图展示了模块内部执行步骤，通过时序图揭露了模块方法调用执行顺序，展现模块的实现思路，并通过关键代码对系统实现进一步补充。最后给出各模块的页面截图。

第六章　系统测试与分析

本章针对基于模糊测试的推荐模型质量分析系统开展全方位的测试工作，以保证系统满足需求设计中提出的功能和非功能性需求。测试工作具体包括：

（1）单元测试。由于系统实现较为复杂，限于篇幅，本文主要针对系统为满足功能性需求所提供的外部接口进行单元测试。测试任务涉及 17 个核心接口。

（2）性能测试。为了验证系统满足非功能性需求，本文给出系统的性能测试结果，主要包括平均响应时间、最大响应时间和错误率。

（3）案例演示。本文提供一个用户从零创建测试任务并查看测试报告的基本案例，展现系统的核心链路和操作流程。

（4）系统有效性验证。本文设计了一个基于 Amazon Music 数据集和 DELF 模型的对比实验，证明了系统能够有效发掘推荐模型历史数据偏见问题，提高测试充分性，并给出系统试运行发现的问题列表。

6.1　测试环境

表 6.1: 测试环境信息

硬件	配置参数
服务器	系统：Ubuntu 20.04.2 LTS 内存：64G CPU： Intel Core i7-9700K CPU @ 3.60GHz (8 核心 16 线程) GPU： Geforce GTX 1080 Ti(11GB) × 2 Vue：2.9.6 Python：3.6.12 Django：3.1.5 scikit-learn：0.20.3 MySQL：8.0.23
开发机	系统：Windows10 64 位 内存：16G CPU： Intel(R) i9-8950HK CPU@2.9GHZ GPU： Geforce GTX 1080(8G) IDE：PyCharm Professional 软件版本与服务器相同

本系统所依赖的测试环境如表6.1所示。单元测试过程在开发机上进行，文

件上传下载位置均为本地磁盘，持久化数据保存在 Django 测试模块自动构建的，与业务数据库隔离的临时数据库中。服务器负责在案例演示和性能测试过程中对测试数据集和扩增数据进行本地存储，采用 MySQL 完成数据持久化，并利用 GPU 支持模糊测试流程，而开发机在这一过程中负责进行前端页面渲染，用户操作交互，数据集/模型文件提供和读取文件编写。

6.2　单元测试

单元测试的目的是保证系统每个模块的执行符合开发预期，因此可以尽早地在开发过程中发现系统缺陷。单元没有固定的划分方式，需要测试人员根据系统规模和模块功能进行人为划分。考虑到系统内部类和方法较多且本文篇幅有限，因此本系统的测试单元划分基于核心功能模块，并进行较为粗粒度的内部划分，单个测试单元中可能包含多个内部方法或类。单元测试过程依赖 Django

表 6.2: 测试环境信息

测试方法	单元说明
test_admin_show_user	用户信息读取单元
test_admin_add_user	用户注册单元
test_admin_check_user	用户名密码检查单元
test_get_user_info	带参用户信息读取单元
test_upload	文件上传单元
test_download	文件下载单元
test_add_structure	特征结构添加单元
test_add_task	测试任务新建单元
test_get_task_info_by_id	带参测试任务信息获取单元
test_get_task_info_by_user	带参测试任务信息获取单元
test_get_task_count	任务数量统计单元
test_get_download_info	下载信息获取单元
test_get_models	模型信息获取单元
test_get_datasets	数据集信息获取单元
test_get_structures	结构信息获取单元
test_get_features_by_structure	特征获取单元
test_get_evaluation_info	评估指标获取单元

框架和其自带的 manage.py 命令行工具。在 test.py 内部完成单元测试代码编写，且保证测试方法命名以 test_ 开头后，Django 内置的单元测试模块发现机制将会完成测试方法扫描和单元测试执行。系统单元测试任务如表6.2所示，其中 test_admin_show_user、test_admin_add_user 和 test_admin_check_user 涉及实现用户登录模块的三个核心单元：用户数据新增、用户数据拉取和密码检查。

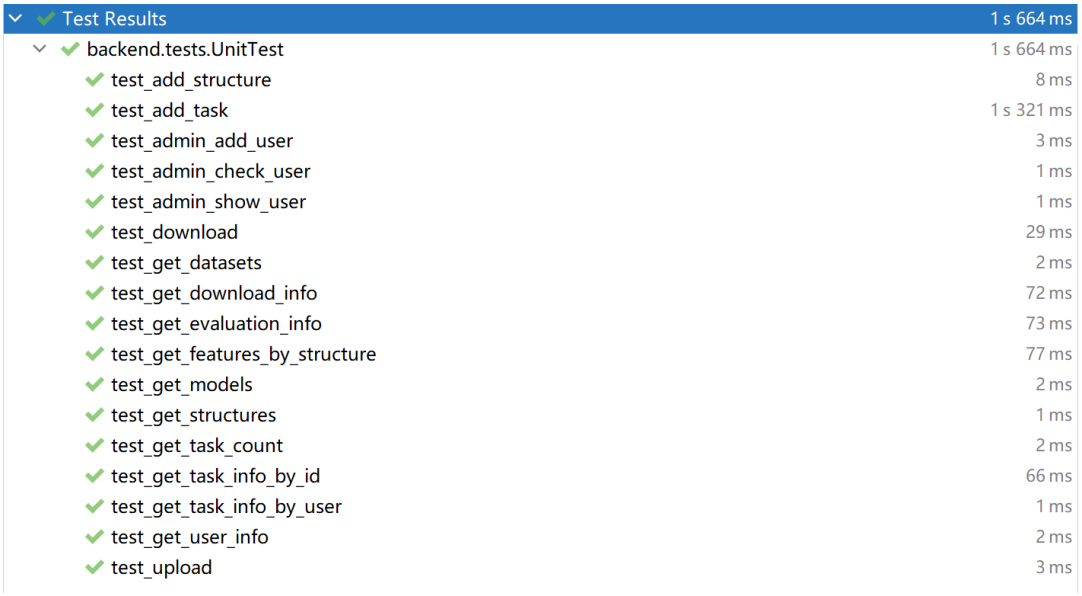

图 6.1: 系统单元测试结果

test_upload 包含所有类型文件的合法性检查和本地存储方法的检查，test_download 检查下载模块的有效性。test_add_structure 和 test_add_task 综合了大量类和方法，测试了特征添加和任务创建过程的有效性，包括特征关联性处理方法检查、用例生成方法检查、用例持久化方法检查等。test_get_task_info_by_id 和 test_get_task_info_by_user 检查了两种获取测试任务信息方法的实际有效性。test_get_task_count 检查了任务统计方法是否能够有效转化时间数据并准确统计出处于时间段内的记录数。test_get_evaluation_info 检查指标信息查询和返回结果的完整性，以及返回数据格式的合法性。

上述单元测试的测试结果如图6.1所示，其中 test_add_task 由于涉及复杂的执行流程，包括持久化、特征构建、数据扩增、数据更新等一系列复杂操作，尽管测试过程未调用具体模型并生成评估指标，只涉及了数据集分析和数据扩增，且数据扩增规模较小，仍然需要相对较长的测试时间。

6.3　性能测试

基于第三章非功能需求的性能部分，表6.3给出系统对应的界面操作响应时间。页面操作依托分析工具 Fiddler，并对文件上传、特征结构创建、对系统的界面响应时间进行测试，分别对文件上传反馈、任务创建、特征构建、下拉框数据响应、报告跳转渲染和数据下载进行各 200 次操作，响应时间和错误率的统计结果符合需求分析规定的性能需求。

表 6.3: 性能测试结果

功能模块	平均响应时间	最长响应时间	错误率
文件上传反馈	0.857s	0.959s	0%
任务创建反馈	0.221s	0.286s	0%
特征结构创建	0.371s	0.624	0%
下拉框数据获取	0.463s	0.522s	0%
报告跳转	0.735s	0.821s	0%
数据下载	0.655s	0.741s	0%

6.4　案例演示

本节将给出用户如何从上传数据集和读取文件开始，从零构建一次测试任务并最终得到测试报告的操作流程演示。演示基于公开数据集 Amazon Music[1]。

6.4.1　文件上传

图 6.2: 上传数据集、数据集读取文件和模型

如图6.2所示，用户首先向系统上传测试所需的初始文件，包括原始数据集、数据集操作文件（包含数据集结构和读取方法实现）和模型文件。考虑到原始数据集较大，为了演示方便，本文采用的 Amazon Music 数据集是对原始数据集进行人工采样后得到的子集。演示数据集与原始数据集除规模外保持一致，均包含两个文件：（1）meta_Digital_Music.json 为特征元数据，采用键值对的形式保存特征名和对应特征值。（2）ratings_Digital_Music.csv 为排名数据，每条样本包含四个属性，依次为 item ID、User ID、排名（rating）和时间戳（timestamp）。

[1]http://jmcauley.ucsd.edu/data/amazon/

amazon_music_reader.py 文件为相应的数据集读取文件，实现了 create_dataset() 方法，用于对数据集进行结构分析和字符校验等预处理。读取文件需要用户完成编写，且需要满足编写格式要求。model_test 为本地训练的一个演示模型，为了保证演示效果，刻意降低了模型质量。图6.2中文件名右侧的绿钩表示文件上传成功并符合系统要求的格式和前置条件。

6.4.2　特征结构创建

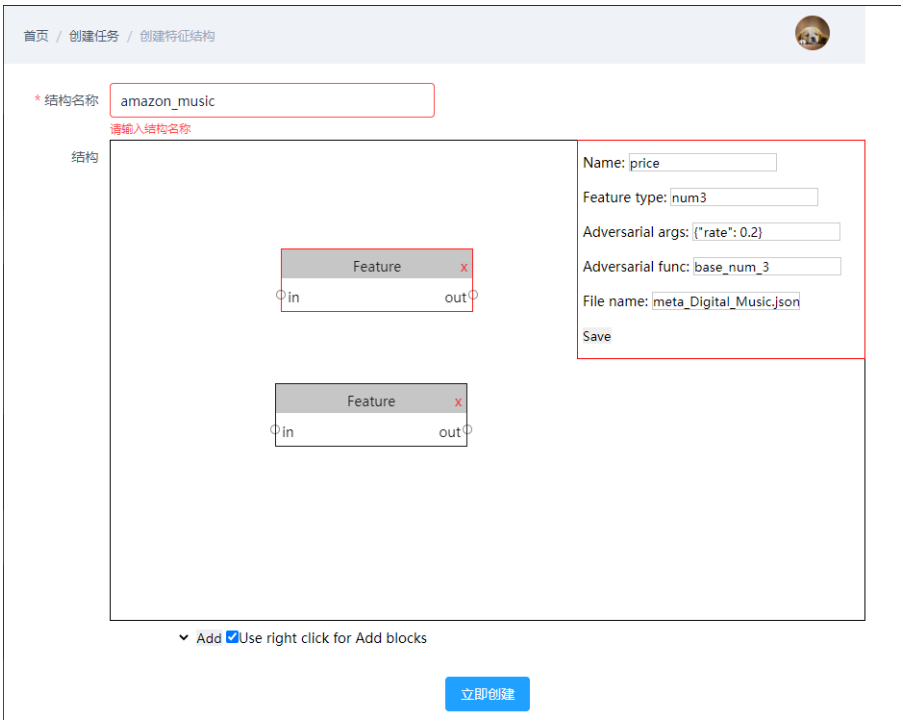

图 6.3: 构建特征结构

如图6.3所示，用户依据测试需要，提供数据集中参与测试任务的特征信息与特征关系。由于 Amazon Music 数据集中没用明确的关联特征，演示案例选择了物品售价 price 和排名 rating 两个核心特征，并进行特征参数配置。

用户通过右击屏幕或特征区域底部的 Add 按钮新增特征信息块。特征块中的 Feature 代表这是一个基本特征，in 和 out 用于关联，在演示案例不需要采用二者进行特征关联连线，同时在右边红框中进行特征信息填充，每个特征对应信息填充完成后，用户点击 save 后完成特征信息保存，保存确认无误后即可点击立即创建，服务器进行特征数据解析和持久化存储。

price 特征为 num3 类型，设置变异概率参数 *rate* 为 0.6，设置变异方法 base_num_3，设置相关文件为）meta_Digital_Music.json。

rating 特征为 num4 类型，对抗参数包括间隔 *gap*、变异概率 *rate* 和取值边界 *bound*，分别设置为 1.0、0.8 和 0.0,5.0。设置变异方法 base_num_4，关联文件为 ratings_Digital_Music.csv。

6.4.3　测试任务创建

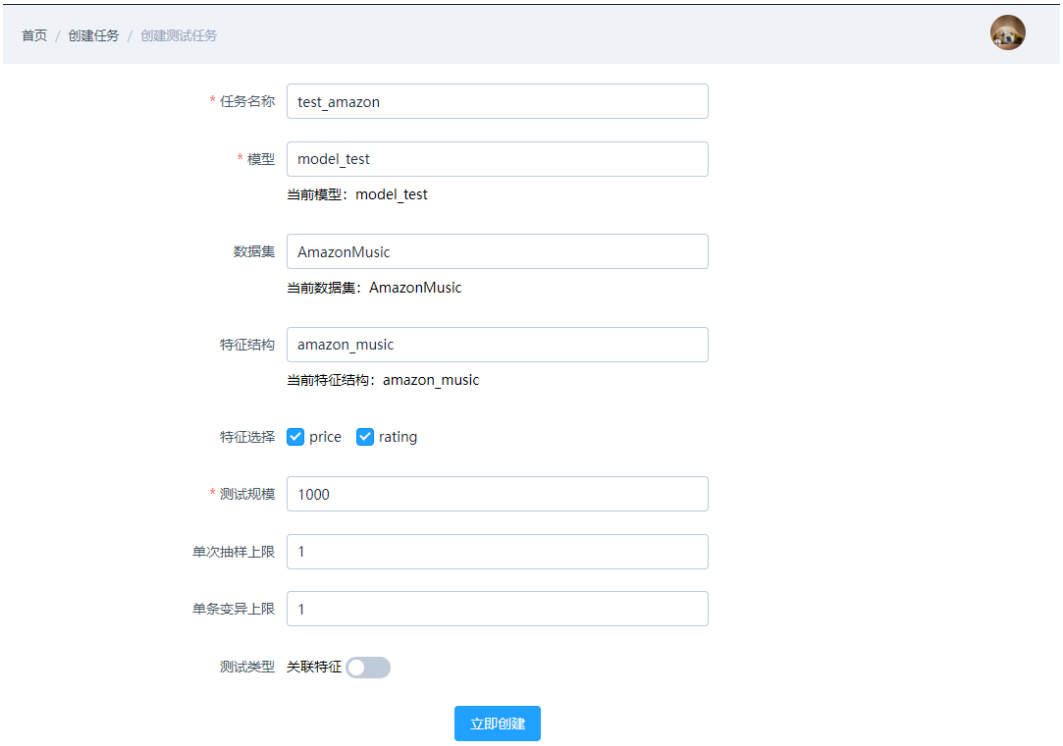

图 6.4: 新建测试任务

当完成文件上传和特征结构创建后用户点击“新建测试任务”开始测试流程。如图6.4所示，用户首先输入任务名称 test_amazon。接着，用户通过下拉框选择待测模型 model_test、数据集 Amazon Music 以及特征结构 amazon_music。确定特征结构 amazon_music 后，系统读取持久化数据，获取所用用户预设的特征，并以复选框组合的形式展示，展示信息为特征名称。复选框中用户必须勾选至少一个特征参与测试，案例中同时勾选了 price 和 rating，意味着变异过程将会兼顾二者。然后用户需要填写测试控制参数，测试规模为 1000 表示生成 1000 条新测试用例，单次抽样上限为 1 表示变异过程中每次迭代从原始数据集中抽取的用例条数为 1 条，单条变异上限为 1 表示单条测试用例中被修改的特征条数最多为 1。同时，用户选择变异过程是否考虑特征间的关联，由于案例中 price

和 rating 特征无相关性，因此设定测试模式为不考虑关联特征模式。

当完成上述信息填写后，点击立即创建提交测试任务。服务器接收请求后依次完成测试用例持久化信息新建、测试用例变异、模型执行、指标计算、持久化记录中的状态信息更新，并存储扩增用例到服务器本地。

6.4.4　测试任务查看

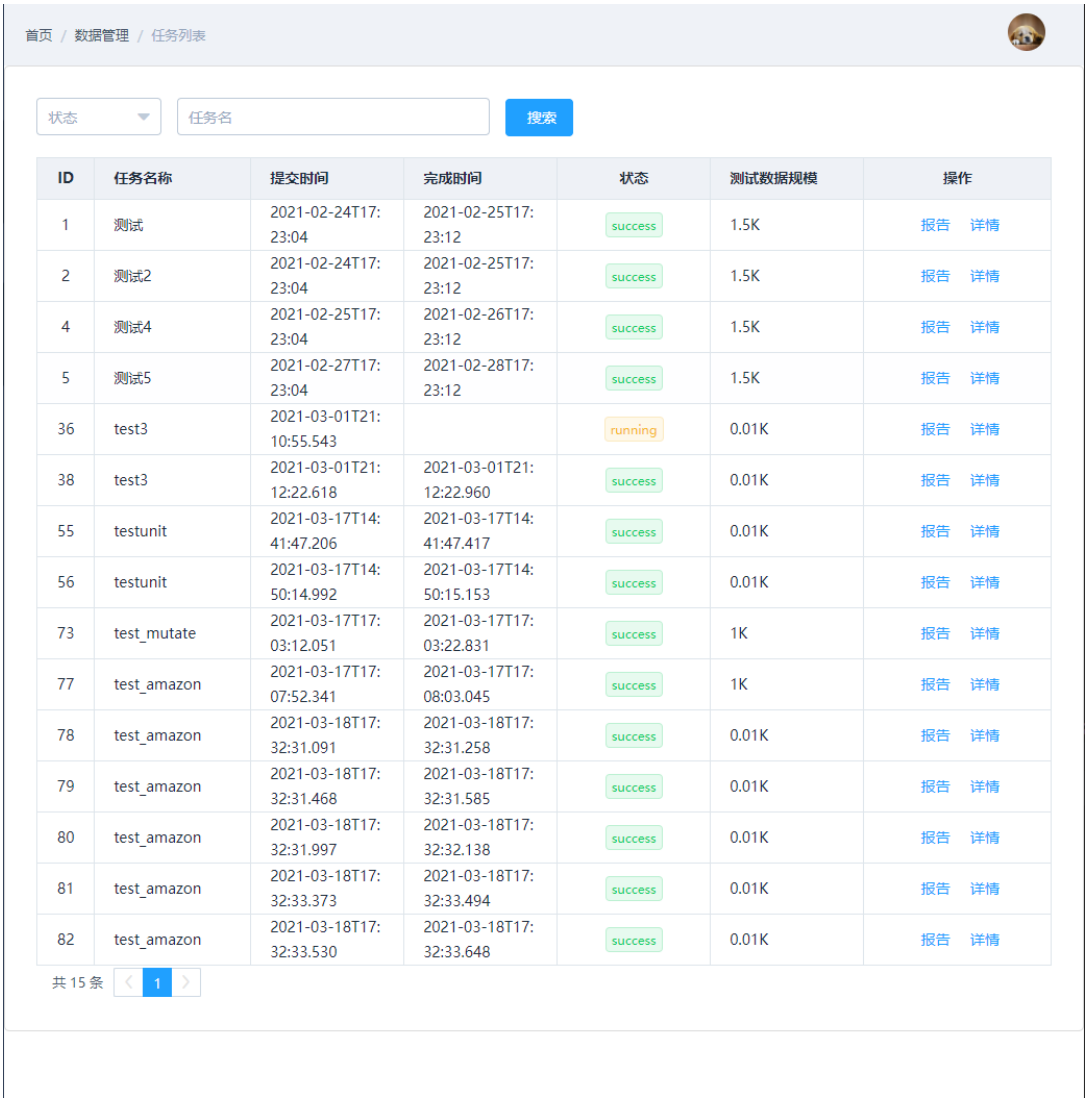

图 6.5: 测试任务查看

ID 为 77 的测试任务即为上述流程新建的测试任务，目前处于已完成状态。测任务列表中给出了测试任务的基本信息，并支持从任务列表进行报告查看和任务详情查看。任务详情查看提供了任务依赖和执行过程的详细信息，除了列

表中已提供的信息外，还包括任务的测试模型信息、数据集信息、任务流程信息、下载信息等一系列额外数据，供用户进行查看。点击报告后即可跳转到用于评估指标和图表展示的报告页面。

6.4.5　评估报告查看

图 6.6: 评估报告查看

系统自动生成的评估报告如图6.6所示。报告页面上侧给出了模糊测试任务相关信息，显示测试任务耗时 11 秒，测试规模为 1K。左侧给出量化指标的计算结果，包括 AUC、精准率、召回率、F1 和 MAP 和热度值。右侧给出的图表分别是 P-R 曲线、t-SNE 降维散点图和热度分桶。通过查看报告可知，一方面案例模型的精准率、召回率和 AUC 值正常，但 F1 和 MAP 值较低，表明模型存在精排质量缺陷。另一方面热度值偏低，热度分桶不均匀，意味着模型推荐长尾现象严重，推荐丰富性不足。除此之外，t-SNE 降维结果散乱，无聚合性，意味着模型存在特征工程质量缺陷。

6.4.6　测试数据下载

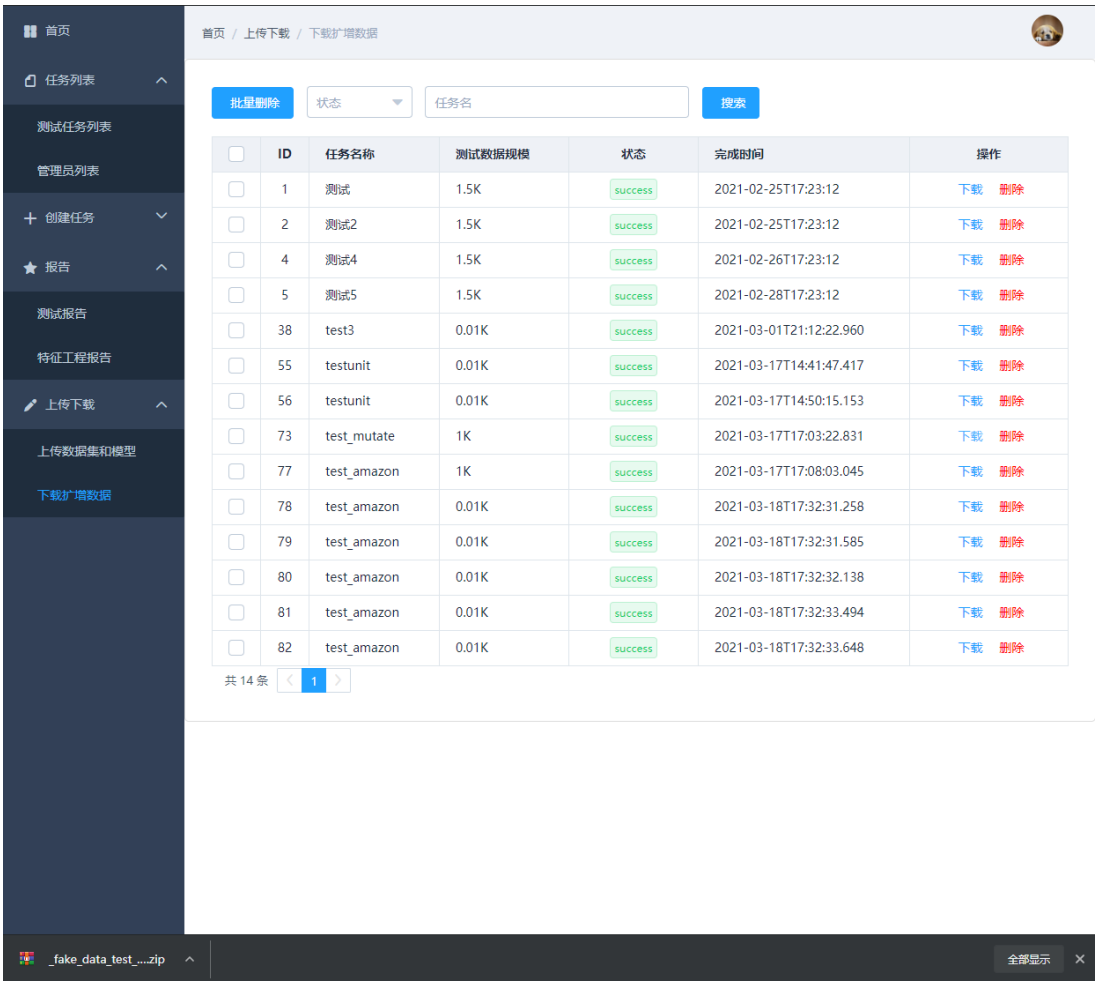

图 6.7: 测试数据下载

测试数据下载结果如图6.7所示。为了演示效果，并未采用搜索框进行定位。列表中提供的是所有已经成功的测试任务，点击下载后会将扩增测试数据进行打包，并下载到用户本地。点击删除后可以将本条下载项从持久化数据中移除，同时移除的还有相应的服务器本地扩增数据文件和持久化的任务数据。但是用户创建的特征结构，数据集文件和模型文件将不会被删除。

6.5　系统有效性验证

本节将给出一组对比实验，分别采用原始数据集和模糊测试过程生成的扩增数据集作为测试数据，通过分析多个推荐规模下，推荐模型的质量评估指标差异，证明本系统能够揭露原始数据集忽略的历史偏见问题，从而提高测试充

分性。同时本节将给出系统在蚂蚁集团试运行过程中，多个应用场景下发掘的推荐模型缺陷的相关记录。

6.5.1　实验设置

数据集。本实验采用公开数据集 Amazon Music[2]，该数据集涉及 1835 名用户的交互信息，包含物品描述记录 41488 条，用户物品交互记录 75932 条。数据集包含两个主要文件，meta_Digital_Music.json 文件包含了物品特征信息，包括物品编号、价格、相似物品列表、物品类型等，ratings_Digital_Music.csv 包含交互信息，每条记录由用户编号、物品编号、用户评分和时间戳信息构成。

待测模型。本实验采用的测试模型是由 Cheng[41] 等人于 2018 年提出的推荐模型 DELF。该隐语义（deep latent factor）模型创新性地采用了基于双重 Embedding 推理的深度学习模型架构，能够更好地利用用户与物品的交互信息，提高推荐质量。本实验采用该模型基于原始数据集的执行结果作为基线。

模糊测试策略。本实验针对 Amazon Music 数据集中物品信息和交互信息进行调整，生成了 10000 条扩增数据。price 特征包含了物品价格信息，本实验将其划分为连续型浮点数，设定舍入精度 2，并采用系统内置的连续型浮点数变异方法。also_bought 特征包含了购买该物品用户的其他购买物品列表，本实验将其划分为列表型，并采用内置方法随机对部分特征值进行列表扩充和列表内容替换。rating 包含用户评分信息，本实验将其划分为离散型浮点数，采用系统内置的离散型浮点数变异方法，并控制变异程度不超过 1 分（最高 5 分，最低 1 分）。ItemID 代表用户购买的物品编号，本实验将其划分为具有待选集且集合元素不可穷举的字符串类型，并在内存中缓存其他用户的物品交互信息，通过内置变异方法扩增某一具体用户的物品交互记录。

评估指标。本实验采用的评估指标包括 AUC，Recall，F1 和 MAP。同时本实验还采用系统提供的热度值 Hot 对推荐结果的多样性进行度量，阈值为 100。为了保证实验的充分性，更好地体现模糊测试方法的测试效果，本实验给出了 Top5、Top10 和 Top20 条件下基线和执行扩增数据集的执行结果。实验采用 Origin@N 表示在 TopN 条件下 DELF 模型基于原始数据集的执行结果，采用 Fuzz@N 表示在 TopN 条件下 DELF 模型基于扩增数据集的执行结果。

测试环境。本实验的硬件环境基于 Intel(R) i9-8950HK CPU 和 Geforce GTX 1080(8G) GPU，内存 16G。为了保证模型稳定执行，测试采用的 TensorFlow 版本为 1.13.1，Cython 版本为 0.29.6，指标计算依赖的 scikit-learn 版本为 0.20.3。

[2]http://jmcauley.ucsd.edu/data/amazon/

6.5.2　对比分析

表 6.4: DELF 模型测试结果

	AUC	Recall	F1	MAP	Hot
Origin@5	0.2253	0.3019	0.1006	0.2094	0.0825
Fuzz@5	0.1654	0.2371	0.0790	0.1490	0.0696
Origin@10	0.2806	0.3526	0.0641	0.2162	0.0630
Fuzz@10	0.2214	0.3020	0.0549	0.1576	0.0211
Origin@20	0.3327	0.4087	0.0389	0.2200	0.0469
Fuzz@20	0.2816	0.3668	0.0349	0.1622	0.0191

实验结果如表6.4所示。通过分析同一测试数据集，不同推荐规模的实验结果可知，随着向用户单次提供的推荐物品数增加，物品获得的曝光频率逐渐提高，用户可选范围扩大，因此召回率和 AUC 稳定提升，且热度不断降低，表明推荐丰富性有所提升。同时，由于用户可选物品增多，整体推荐有效性提升，导致 MAP 值略微上升。除此之外，由于用户交互行为有限，推荐物品的增加导致用户交互比重降低，因此 F1 值不断下降。

在推荐规模一定的情况下，分别采用原始数据集和扩增数据集进行验证，模型表现出现了显著差异。一方面，相比于原始数据，扩增数据集内部存在交互信息调整和物品特征信息的细微变化，导致 Recall、AUC、F1 和 MAP 指标值均有不同程度的下降，反映推荐模型无法有效应对这一调整，说明了模型存在针对原有数据集的历史数据偏见问题。同时，由于扩增数据集包含了更多用户物品的交互信息，推荐结果的多样性有所提升。这一对比结果说明了模糊测试方法能够有效地揭露推荐模型的历史数据偏见问题，提高测试充分性。

6.5.3　试运行结果分析

本系统在蚂蚁集团内部试运行过程中，由数据算法质量部进行部署与维护，并用于对支付宝线主要应用场景下的推荐质量进行离线评估。8 个具体场景包括三月大促、城际服务、智能 UI、新春五福、首页推荐、搜索服务、搜索底纹词和福卡推荐，每个场景下包含多个推荐模型或推荐模型的多轮迭代。

系统在试运行过程中发现的推荐模型缺陷如表6.5所示。系统发现 12 个模型存在特征工程质量缺陷，该缺陷通过 t-SNE 降维散点图分布异常透出。系统通过量化指标发现 8 个模型存在质量缺陷，具体包括 1 个城际服务模型召回率较低，1 个智能 UI 模型精准率较低，1 个智能 UI 模型热度分桶不均匀导致长尾严重，1 个首页模型精准率较低，1 个首页模型热度值异常推荐多样性不足，1 个

表 6.5: 系统试运行结果

模型应用场景	异常指标	缺陷分析
三月大促	t-SNE 混乱	模型特征工程存在缺陷
城际服务	t-SNE 混乱，Recall 较低	模型判别质量差，特征工程存在缺陷
智能 UI	t-SNE 混乱，Precision 较低	模型推荐质量差，特征工程存在缺陷
智能 UI	热度值异常，热度分桶头部比重过大	模型推荐长尾严重
智能 UI	t-SNE 混乱	特征工程存在缺陷
新春五福	t-SNE 混乱	特征工程存在缺陷
首页推荐	t-SNE 混乱，Precision 较低	模型推荐质量差，特征工程存在缺陷
首页推荐	t-SNE 混乱，热度值异常	模型推荐丰富性差，特征工程存在缺陷
搜索服务	Precision/Recall 较低	模型推荐质量差
搜索服务	t-SNE 混乱	特征工程存在缺陷
搜索服务	t-SNE 混乱，AUC 较低	模型精排较差，特征工程存在缺陷
福卡推荐	t-SNE 混乱	特征工程存在缺陷
搜索底纹词	t-SNE 混乱，热度值异常	模型推荐丰富性差，特征工程存在缺陷
搜索底纹词	t-SNE 混乱	特征工程存在缺陷

搜索服务模型精准率和召回率均偏低，1 个搜索服务模型 AUC 较低，1 个搜索底纹词模型热度值异常推荐多样性不足。

6.6　本章小结

本章首先给出系统的单元测试和性能测试结果数据，证明了系统满足了非功能需求。然后，本章设计并详细分析了一个完整测试案例，案例包含从文件上传到测试数据下载和报告查看的整套流程，展示了系统能够有效帮助开发者进行推荐模型质量评估。最后，本章给出系统有效性验证。一方面，本章基于 DELF 模型设计了一组对比实验，证明了本系统基于模糊测试方法能够发掘推荐模型的历史数据偏见问题，提高推荐模型的测试充分性。另一方面，本章给出了本系统在蚂蚁集团内部试运行过程中，针对 8 个主要业务场景，通过量化指标发掘了 8 个推荐模型存在推荐结果丰富性不足等质量缺陷，通过 t-SNE 可解释性分析揭露了 12 个推荐模型存在特征工程质量问题，展现了系统的实用性。

第七章　总结与展望

7.1　总结

推荐模型相关技术的快速发展，使得互联网公司能够为用户提供更优质的推荐服务，在优化用户体验的同时，为公司带来了用户流量增长或商品交易量增加等诸多效益。因此，保障推荐系统的模型质量至关重要。现有的推荐模型测试方法存在忽视历史数据偏见和测试指标单一等诸多问题，而新颖的基于对抗攻击的推荐模型测试方法由于测试流程复杂，测试成本过高难以被广泛使用。

为了克服上述问题，发掘推荐模型缺陷，提高测试充分性，本文设计并实现了基于模糊测试的推荐模型质量分析系统。该系统通过特征结构建立特征信息与测试数据的内在联系，并由开发者指定测试数据中不同属性值所对应的特征类型和特征变异方法，以及不同特征间的关联关系。基于特征结构，系统采用模糊测试方法，利用开发者提供的数据集和丰富的内置特征变异方法库，快速高效地实现测试数据扩增，提高了测试充分性。系统收集扩增数据执行结果，计算不同维度的质量评估指标并进行可解释性分析，以此给出质量评估报告。

本文首先介绍了本系统的课题背景与研究意义，阐述了推荐模型质量分析工作的研究现状与不足，并对系统测试对象，系统所采用的模糊测试方法，以及系统实现所依赖的前后端技术框架和算法包进行说明。接着，本文基于系统涉众分析，给出系统的功能性需求和非功能性需求，以及系统详细用例描述，并通过提供系统的功能模块划分与 4+1 视图进行补充。基于系统实体关系，本文给出了系统的持久化设计。然后，本文介绍了系统内置的特征划分依据和变异方法，以及系统内置的评估指标定义和可解释性分析过程。针对系统每个功能模块，本文分别给出流程图与时序图，并与关键代码相结合，阐述了模块的实现细节，同时给出实现截图进行补示。最后，为了保障系统的有效运行，本文给出了系统的单元测试和性能测试结果，并演示了一则使用案例。同时，为了验证系统有效性，本文基于 Amazon Music 数据集设计了一组对比实验，并给出了系统试运行过程中发现的推荐模型质量缺陷的详细信息。

系统目前已在蚂蚁集团内部试运行，基于热度等评估指标，完成了 8 个主要应用场景下推荐模型的质量评估。系统通过量化指标发掘了 8 个模型存在推荐结果丰富性不足等质量缺陷，通过 t-SNE 可解释性指标揭露了 12 个模型存在特征工程质量问题，为开发者提供了切实帮助。

7.2 展望

本文初步实现了基于模糊测试的推荐模型质量分析系统，系统在功能丰富性和使用方便性上仍有不足之处需要改进：

（1）尽管目前提供了用户自定义特征类型和变异方法，但尚未支持评估指标自定义，后续将支持用户扩展评估指标，并自动化调整评估报告前端布局。

（2）目前不支持数据集结构自动化分析，仍需要用户手动编写并上传数据集读取代码。后续将提供读取规则自动化构建功能，系统向用户提供可交互页面，通过页面操作的方式完成数据集读取构建，减少用户使用成本。

（3）由于模型结构自动化调整颇为复杂，多轮模型重训练成本较高，系统尚未支持采用对抗攻击方法测试推荐模型。后续系统将支持对抗攻击测试流程。

参考文献

[1] SHAPIRA B, ROKACH L, FREILIKHMAN S. Utilizing Facebook Single and Cross Domain Data for Recommendation Systems[J]. User Modeling and User-Adapted Interaction, 2013.

[2] HSU F-M, LIN Y-T, HO T-K. Design and Implementation of An Intelligent Recommendation System For Tourist Attractions: The Integration of EBM Model, Bayesian Network and Google Maps[J]. Expert Systems with Applications, 2012.

[3] ZHOU R, KHEMMARAT S, GAO L. The Impact of YouTube Recommendation System on Video Views[C] // Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. 2010 : 404 – 410.

[4] SINGEL R. Netflix Cancels Recommendation Contest after Privacy Lawsuit[J]. Wired Magazine, 2010.

[5] WANG J, HUANG P, ZHAO H, et al. Billion-Scale Commodity Embedding for E-commerce Recommendation in Alibaba[C] // Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018 : 839 – 848.

[6] HUANG Y, CUI B, ZHANG W, et al. Tencentrec: Real-Time Stream Recommendation in Practice[C] // Proceedings of the 41st ACM SIGMOD International Conference on Management of Data. 2015 : 227 – 238.

[7] KARLGREN J. An Algebra for Recommendations: Using Reader Data as A Basis for Measuring Document Proximity[J]. Stockholm University, 1990.

[8] COVINGTON P, ADAMS J, SARGIN E. Deep Neural Networks for Youtube Recommendations[C] // Proceedings of the 10th ACM Conference on Recommender Systems. 2016 : 191 – 198.

[9] WANG Z. 深度学习推荐系统 [M]. [S.l.] : 电子工业出版社出版, 2020.

[10] CHENG H-T, KOC L, HARMSEN J, et al. Wide & Deep Learning for Recommender Systems[C] // Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. 2016 : 7 – 10.

[11] FRASER G, ARCURI A. Evosuite: Automatic Test Suite Generation for Object-Oriented Software[C] // Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering. 2011 : 416 – 419.

[12] KICIMAN E, LIVSHITS B. AjaxScope: a Platform for Remotely Monitoring the Client-Side Behavior of Web 2.0 Applications[J]. ACM SIGOPS Operating Systems Review, 2007.

[13] JIANG H, CHEN X, HE T, et al. Fuzzy Clustering of Crowdsourced Test Reports for Apps[J]. ACM Transactions on Internet Technology, 2018.

[14] LI L, CHU W, LANGFORD J, et al. Unbiased Offline Evaluation of Contextual-Bandit-Based News Article Recommendation Algorithms[C] // Proceedings of the 4th ACM International Conference on Web Search and Data Mining. 2011 : 297 – 306.

[15] RADLINSKI F, CRASWELL N. Optimized Interleaving for Online Retrieval Evaluation[C] // Proceedings of the 6th ACM International Conference on Web Search and Data Mining. 2013 : 245 – 254.

[16] SZEGEDY C, ZAREMBA W, SUTSKEVER I, et al. Intriguing Properties of Neural Networks[J]. arXiv preprint arXiv:1312.6199, 2013.

[17] GOODFELLOW I J, SHLENS J, SZEGEDY C. Explaining and Harnessing Adversarial Examples[J]. arXiv preprint arXiv:1412.6572, 2014.

[18] MOOSAVI-DEZFOOLI S-M, FAWZI A, FROSSARD P. Deepfool: A Simple and Accurate Method to Fool Deep Neural Networks[C] // Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition. 2016 : 2574 – 2582.

[19] ODENA A, OLSSON C, ANDERSEN D, et al. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing[C] // Proceedings of the 36th International Conference on Machine Learning. 2019 : 4901 – 4911.

[20] HE X, HE Z, DU X, et al. Adversarial Personalized Ranking for Recommendation[C] // Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. 2018 : 355 – 364.

[21] TANG J, DU X, HE X, et al. Adversarial Training Towards Robust Multimedia Recommender System[J]. IEEE Transactions on Knowledge and Data Engineering, 2019.

[22] WANG J, YU L, ZHANG W, et al. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models[C] // Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 2017 : 515 – 524.

[23] GUO J, JIANG Y, ZHAO Y, et al. DLFuzz: Differential Fuzzing Testing of Deep Learning Systems[C] // Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2018 : 739 – 743.

[24] LUO W, CHAI D, RUN X, et al. Graph-Based Fuzz Testing for Deep Learning Inference Engine[J]. arXiv preprint arXiv:2008.05933, 2020.

[25] MILLER B P, FREDRIKSEN L, SO B. An Empirical Study of the Reliability of UNIX Utilities[J]. Communications of the ACM, 1990.

[26] GOLDBERG D, NICHOLS D, OKI B M, et al. Using Collaborative Filtering to Weave an Information Tapestry[J]. Communications of the ACM, 1992.

[27] LINDEN G, SMITH B, COM J Y A. Industry Report: Amazon.com Recommendations: Item-To-Item Collaborative Filtering[C] // Proceedings of the 7th IEEE Internet Computing. 2003 : 76 – 80.

[28] KONSTAN J A, MILLER B N, MALTZ D, et al. Grouplens: Applying Collaborative Filtering to Usenet News[J]. Communications of the ACM, 1997.

[29] SARWAR B, KARYPIS G, KONSTAN J, et al. Item-Based Collaborative Filtering Recommendation Algorithms[C] // Proceedings of the 10th International Conference on World Wide Web. 2001 : 285 – 295.

[30] KOREN Y, BELL R, VOLINSKY C. Matrix Factorization Techniques for Recommender Systems[J]. Computer, 2009.

[31] LEMARÉCHAL C. Cauchy and the Gradient Method[J]. Doc Math Extra, 2012.

[32] RICHARDSON M, DOMINOWSKA E, RAGNO R. Predicting Clicks: Estimating the Click-Through Rate for New Ads[C] // Proceedings of the 16th International Conference on World Wide Web. 2007 : 521 – 530.

[33] RENDLE S. Factorization Machines[C] // Proceedings of the 10th IEEE International Conference on Data Mining. 2010 : 995 – 1000.

[34] HE X, PAN J, JIN O, et al. Practical Lessons from Predicting Clicks on Ads at Facebook[C] // Proceedings of the 8th International Workshop on Data Mining for Online Advertising. 2014 : 1 – 9.

[35] GAI K, ZHU X, LI H, et al. Learning Piece-Wise Linear Models from Large Scale Data for Ad Click Prediction[J]. arXiv preprint arXiv:1704.05194, 2017.

[36] WANG R, FU B, FU G, et al. Deep & Cross Network for Ad Click Predictions[C] // Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017 : 1 – 7.

[37] GUO H, TANG R, YE Y, et al. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction[J]. arXiv preprint arXiv:1703.04247, 2017.

[38] HE X, CHUA T-S. Neural Factorization Machines for Sparse Predictive Analytics[C] // Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2017 : 355 – 364.

[39] XIAO J, YE H, HE X, et al. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks[J]. arXiv preprint arXiv:1708.04617, 2017.

[40] KRUCHTEN P B. The 4+ 1 View Model of Architecture[J]. IEEE Software, 1995.

[41] CHENG W, SHEN Y, ZHU Y, et al. DELF: A Dual-Embedding Based Deep Latent Factor Model for Recommendation.[C] // Proceedings of the 27th International Joint Conference on Artificial Intelligence. 2018 : 13 – 19.

简历与科研成果

基本情况 曹可凡，男，汉族，1997 年 8 月出生，江苏省盐城市人。

教育背景

2019.9 — 2021.6 南京大学软件学院 硕士

2015.9 — 2019.6 南京工业大学计算机科学与技术学院 本科

论文

1. Wang J, **Cao K**, Fang C, et al. FDFuzz: Applying Feature Detection to Fuzz Deep Learning Systems[J]. International Journal of Performability Engineering, 2019, 15(10): 2675.

专利

1. 房春荣、**曹可凡**、吕军、章许帆、沪胜浩、顾逸飞”一种针对深度学习算子运算精度的差分模糊测试工具“，申请号：2020104872014，已受理。
2. 房春荣、**曹可凡**、顾逸飞、章许帆、沪胜浩、吕军”一种基于模糊测试的深度学习算子测试工具“，申请号：2020104871651，已受理。
3. 房春荣、**曹可凡**、吕军、章许帆、顾逸飞、沪胜浩”一种基于 Fuzz 的人工神经网络测试方法“，申请号：2020104871647，已受理。

致　　谢

初写致谢，耳机里正好唱到“将往事留在风中”，心中一阵不真实的，却又是久违的坠落现实的悸动。直到那时才真真实实地体会到，是到要毕业的时候了。与过去的数次分别一样，心中既是欣喜，为同学们学有所成，都有着光明的未来而高兴，又是怅然，只觉得再遇不到这么好的老师和同学，能开拓我的视野，启发我的思考。这次更特殊，这是最后一次的毕业，这是学生时代的谢幕。

记得本科刚毕业那天的炎炎烈日，夏风已裹挟着热浪，催促我赶快跨江回到老校区，因为第二天需要踏上去杭州的列车。在 iSE 实验室的两年，除了学习了很多知识，也在陈老师的带领和安排下去了全国许多城市，开阔了视野。这两年中我经历了第一次坐高铁，第一次坐飞机，第一次在陌生的城市一个人穿行，除了感谢陈老师给我两年的指导和肯定，还要着重感谢这些出差安排，帮助我克服了自己原本对于旅行的担忧和害怕，培养了我独立自主的能力。又想到自己反复被陈老师批评“摸鱼划水”，仍然非常惭愧和遗憾。

校园里的秋风中有桂花香，第一年的秋天在房老师的指导和吕军的帮助下完成算子测试项目。房老师不仅在专利申请和项目开发上给予很多指导，平时生活中亦是关照颇多，现在还记得自己收到房老师送的生日蛋糕时的感动。吕军的代码能力和业务能力要超过我许多，没有他很多实际工作我都难以完成，更重要的是他一直是静静听，然后去传递鼓励和关心，永远是有向上的正能量。

南大第一年冬天的寒风更加凌冽，没想到那年年会结束后与小伙伴们的分别竟是三个季度。新冠疫情改变了中国，改变了世界，也改变了每个南大人的生活轨迹。还好过去了，第二年冬天的年味里混着辛苦完成毕设的汗水味儿。非常感谢章许帆学长在毕设进度上的把控和内容上的指导，以及房老师在我论文内容出现偏差后进行的批评与耐心指正。在他们的帮助下，我才最终重新振奋精神，严格要求自己，找回南大学子应有的精神。

春风中的青草味呼唤着我走出宿舍，告诉我是时候去拿起羽毛球拍，舒展一下沉寂了一个冬天的身躯。非常感谢郭超、孙加辉、张朱佩田能够在给予我课程和实验室工作上的帮助外，还带我一起打羽毛球。同时也要感谢可爱室友曾豪、曹凯博和常辰阳，感谢他们在两年宿舍生活中对我的包容和给我的欢笑。

只觉得每次毕业时都感慨往事如梦，回首看来已是雾里看花，不仅是想不起当时的喜怒哀乐，竟连细节也变得模糊，因此在致谢中多留一些无关紧要的内容，用文字来记录情感的同时，亦致研究生生活。