

自然写教学数据分析与学情诊断系统软件 V1.0

软件著作权鉴别材料（设计说明书）

项目	内容
软件全称	自然写教学数据分析与学情诊断系统软件
软件简称	自然写学情系统
版本号	V1.0
权利人	深圳自然写科技有限公司
开发语言	Python / Java
运行环境	Linux服务器
文档类型	设计说明书
编制日期	2026年2月

目录

- 第一章 软件整体概述
 - 1.1 软件简介与功能综述
 - 1.2 软件用途与适用场景
 - 1.3 运行环境与系统要求
 - 1.4 开发语言与技术规范
 - 1.5 版本说明
- 第二章 系统架构与设计思路
 - 2.1 总体架构设计
 - 2.2 数据仓库架构
 - 2.3 知识图谱设计
 - 2.4 数据模型设计
 - 2.5 接口设计
 - 2.6 安全设计
 - 2.7 部署架构

- 第三章 核心模块功能详细说明
 - 3.1 实时数据采集与ETL模块
 - 3.2 学生个人学情画像模块
 - 3.3 班级与年级学情统计模块
 - 3.4 作业与考试成绩分析模块
 - 3.5 书写能力成长轨迹模块
 - 3.6 错题归因与知识图谱关联模块
 - 3.7 教学效果评估模块
 - 3.8 可视化报表与数据导出模块
 - 3.9 家长端学情推送模块
- 第四章 操作流程与使用步骤
 - 4.1 系统部署与初始化
 - 4.2 数据源配置与接入
 - 4.3 学情报告查看操作流程
 - 4.4 班级学情分析操作流程
 - 4.5 自定义报表配置流程
 - 4.6 异常处理与数据质量保障
- 第五章 与源代码的对应关系
 - 5.1 模块与源代码文件对应表
 - 5.2 核心函数说明
 - 5.3 命名规范
- 附录

第一章 软件整体概述

1.1 软件简介与功能综述

自然写教学数据分析与学情诊断系统软件（以下简称“学情系统”）是自然写互动课堂平台的大数据分析核心组件，负责对学生书写及答题数据进行多维度大数据分析，生成个性化学情诊断报告，为教师、学校管理者、家长提供数据驱动的教学洞察和决策支撑。

学情系统采用数据仓库与实时分析引擎相结合的技术架构，通过Apache Kafka接收实时数据流，Apache Flink进行流式ETL处理，ClickHouse存储分析型数据，Python（Pandas/Scikit-learn）实现诊断算法，Neo4j构建知识点关联图谱，ECharts驱动可视化报表展示。

主要功能模块：

(1) 学生个人学情画像：为每位学生构建多维度的学习画像，涵盖各学科知识点掌握度热力图、学习能力雷达图、书写质量趋势线、进步/退步预警标识等，帮助教师全面了解每位学生的学习状态。

(2) 班级与年级学情统计：以班级或年级为单位，统计作业平均分分布、成绩区间分布（优/良/中/差）、知识掌握薄弱区域、整体书写规范程度等群体指标，辅助教研决策。

(3) 作业与考试成绩分析：对每次作业或考试进行深度分析，包括难度系数（各题得分率）、区分度（高分段和低分段学生在各题上的得分差异）、知识点覆盖矩阵、班级内横向对比等。

(4) 书写能力成长轨迹：以时间轴形式展示每位学生的书写质量历史变化，包括OCR识别准确率趋势（反映字体规范程度）、笔顺正确率趋势、书写速度变化等。

(5) 错题归因与知识图谱：将学生的错误答案与知识图谱中的知识点进行关联，找出知识掌握链条上的断点，推断错误的深层原因（如“应用题不会”的背后是“分数除法掌握不牢”）。

(6) 教学效果评估：基于学生成绩数据评估教师的教学效果，生成教研参考报告，帮助教师识别教学内容中的薄弱环节。

(7) 可视化报表：提供丰富的ECharts图表（折线图、柱状图、雷达图、热力图、桑基图等），支持报告PDF导出。

(8) 家长端推送：定期（每日/每周/每月可配置）将学情摘要报告推送至家长手机，提升家校协同效率。

1.2 软件用途与适用场景

学情系统的核心价值在于将海量的书写数据和答题数据转化为可行动的教学洞察，适用于以下场景：

(1) 教师日常教学参考：教师在布置新作业前查看上一次作业的错题分析，了解哪些知识点学生普遍掌握不好，据此调整本次教学重点。

(2) 家长了解子女学习状况：家长每周收到子女学情推送，看到本周作业完成情况、书写进步或退步情况、知识掌握变化，及时与教师沟通。

(3) 学校管理者教学质量监控：教务主任或校长可查看全校各班级的教学质量横向对比，识别学习成绩显著偏低或进步显著的班级，进行针对性的管理干预。

(4) 教研活动数据支撑：教研组利用学情数据进行集体备课和教研讨论，基于数据客观评估某种教学方法的有效性。

(5) 个性化学习推荐：基于学生的知识掌握画像，为学生推荐有针对性的练习题和学习资源，实现个性化教学。

1.3 运行环境与系统要求

组件	要求
操作系统	Linux（CentOS 7.6+ / Ubuntu 20.04+）
Python版本	Python 3.9+
Java版本	OpenJDK 11+（Flink使用）
Apache Kafka	3.4+（数据接入消息队列）
Apache Flink	1.17+（实时流处理ETL）
ClickHouse	23.x+（OLAP分析型数据库）
MySQL	8.0+（OLTP业务数据）
Neo4j	5.x+（知识图谱数据库）
MongoDB	6.0+（报告快照存储）
Redis	7.0+（实时数据缓存）
最低服务器配置	16核CPU、64GB内存、2TB SSD

ClickHouse集群规格建议（按学校规模）：

学校规模	建议集群规格
小型（≤500学生）	3节点，每节点8核16GB
中型（500–3000学生）	6节点，每节点16核32GB
大型（>3000学生）	12+节点，每节点32核64GB

1.4 开发语言与技术规范

主要开发语言与框架：

模块	语言/框架	说明
数据采集ETL	Java（Apache Flink）	流式数据处理，Kafka消费到ClickHouse写入
诊断算法层	Python（Pandas, Scikit-learn, NetworkX）	学情分析算法、机器学习模型、图谱推理

模块	语言/框架	说明
REST API服务	Python (FastAPI)	对外提供学情数据查询接口
可视化后端	Python (Matplotlib, Pillow)	服务端图表渲染 (PDF报告)
知识图谱操作	Python (py2neo)	Neo4j图数据库访问
报告生成	Python (ReportLab / WeasyPrint)	PDF报告排版生成

1.5 版本说明

版本号	发布日期	说明
V1.0	2026年2月	初始版本, 包含学情画像、班级分析、成绩分析、成长轨迹、知识图谱、报告导出全功能

第二章 系统架构与设计思路

2.1 总体架构设计

学情系统采用Lambda架构思想, 将数据处理分为实时流处理层 (Speed Layer) 和批处理层 (Batch Layer), 保证了系统既能实时响应新数据, 又能对历史数据进行全量深度分析。





2.2 数据仓库架构

数据仓库采用星型模型设计，以学情事实数据为中心，维度表为星芒：

事实表（ClickHouse）：

- `fact_submission`：作业提交事实表，记录每次作业提交的得分、知识点覆盖、耗时等指标
- `fact_writing_quality`：书写质量事实表，记录每次书写的OCR准确率、笔顺得分、质量评分

维度表（MySQL + ClickHouse）：

- `dim_student`：学生维度，含学生ID、姓名、班级、年级、入学年份
- `dim_knowledge_point`：知识点维度，含知识点ID、名称、学科、年级、父级知识点
- `dim_assignment`：作业维度，含作业ID、标题、学科、类型、难度级别
- `dim_date`：日期维度，含年、月、周、日、学期等时间属性

聚合指标表（ClickHouse）：

- `agg_student_weekly`：学生每周聚合指标（周平均分、书写平均分、提交完成率）
- `agg_class_daily`：班级每日聚合指标（当日作业完成率、平均得分）
- `agg_knowledge_mastery`：知识点掌握度聚合（学生×知识点掌握度矩阵，按周更新）

2.3 知识图谱设计

知识图谱使用Neo4j图数据库存储，将学科知识体系建模为有向图：

节点类型：

节点类型	属性	说明
Subject	id, name, grade_level	学科节点（语文、数学、英语等）

节点类型	属性	说明
Chapter	id, name, subject_id, sequence	章节节点
KnowledgePoint	id, name, chapter_id, difficulty	知识点节点
Concept	id, name, description	概念节点（跨章节的通用概念）

关系类型：

关系类型	方向	说明
CONTAINS	Subject→Chapter, Chapter→KnowledgePoint	包含关系
PREREQUISITE	KnowledgePoint→KnowledgePoint	先修关系（学习B需要先掌握A）
APPLIES	KnowledgePoint→Concept	应用关系
SIMILAR	KnowledgePoint↔KnowledgePoint	相似关系（无方向）

典型错题归因查询（Cypher语言）：

```
// 查找学生在某知识点上出错后，可能掌握不牢的前驱知识点
MATCH (kp:KnowledgePoint {id: $error_kp_id})
MATCH path = (prereq:KnowledgePoint)-[:PREREQUISITE*1..3]->(kp)
WHERE prereq.id IN $student_weak_kp_list
RETURN prereq.name AS missing_prerequisite,
        length(path) AS hop_distance
ORDER BY hop_distance ASC
LIMIT 5
```

2.4 数据模型设计

学情事实表（ClickHouse – fact_submission）：

```
CREATE TABLE fact_submission (
    submission_id    UInt64,
    student_id       UInt64,
    assignment_id     UInt64,
    class_id         UInt32,
    school_id        UInt32,
    subject          LowCardinality(String),
    grade            UInt8,
    submit_time       DateTime,
    total_score       Float32,
    max_score        Float32,
    score_rate       Float32,           -- 得分率 = total_score / max_score
    writing_score      Float32,         -- 书写质量分（若有）
    stroke_order_score Float32,        -- 笔顺分（若有）
)
```

```
time_spent_seconds UInt32,      -- 答题用时 (秒)
is_late_submit UInt8           -- 是否迟交 (0/1)
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(submit_time)
ORDER BY (school_id, class_id, student_id, submit_time);
```

知识点掌握度宽表 (ClickHouse – agg_knowledge_mastery):

```
CREATE TABLE agg_knowledge_mastery (
  student_id      UInt64,
  knowledge_point_id UInt32,
  subject         LowCardinality(String),
  grade          UInt8,
  mastery_score   Float32,  -- 掌握度分 (0-100), 综合近期得分率计算
  error_count     UInt16,   -- 该知识点累计出错次数
  last_correct_date Date,   -- 最近一次答对的日期
  last_error_date Date,    -- 最近一次答错的日期
  trend          Int8,      -- 近两周趋势 (-1下降/0持平/1上升)
  updated_at      DateTime
) ENGINE = ReplacingMergeTree(updated_at)
ORDER BY (student_id, knowledge_point_id);
```

2.5 接口设计

主要API接口 (FastAPI):

接口名称	HTTP方法	路径	说明
学生画像	GET	/api/v1/profile/student/{id}	获取学生完整学情画像数据
班级学情	GET	/api/v1/report/class/{id}	班级学情统计 (支持按时间范围过滤)
年级对比	GET	/api/v1/report/grade/{school_id}	年级内各班级横向对比
错题分析	GET	/api/v1/error/analysis/{student_id}	学生错题归因与薄弱知识点
知识掌握热力图	GET	/api/v1/heatmap/{student_id}	学生知识点掌握度热力图数据
成长轨迹	GET	/api/v1/growth/{student_id}	书写能力和成绩成长时序数据
作业深度分析	GET	/api/v1/assignment/analysis/{id}	单次作业的难度/区分度/知识点分析
报告导出	POST	/api/v1/report/export	生成PDF报告, 返回下载URL
家长推送预览	GET	/api/v1/push/preview/{student_id}	预览即将发送给家长的学情摘要

接口名称	HTTP方法	路径	说明
手动触发推送	POST	/api/v1/push/trigger	手动触发学情报告推送（教师操作）

2.6 安全设计

数据权限控制：

学情数据涉及学生隐私，系统严格按照角色执行数据权限隔离：

- 超级管理员：可查看全平台所有数据，包括跨校数据
- 学校管理员：可查看本校范围内所有班级和学生数据
- 教师：仅可查看自己担任教师的班级的学生数据，不可跨班查看
- 家长：仅可查看自己子女的学情数据
- 学生：仅可查看自己的学情数据

权限校验在FastAPI中间件层实现，每个API请求均通过JWT解析用户身份和角色后，与请求的资源（student_id/class_id）进行权限比对。

数据脱敏：

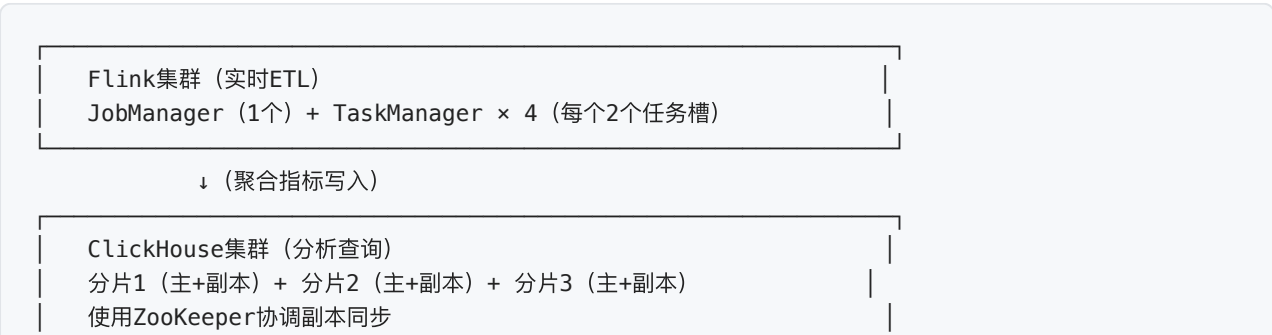
导出的报告文件中，学生姓名默认保留全名；在系统管理界面以外的公共展示场景（如大屏投影）中，学生姓名自动脱敏为"张*"格式，保护学生隐私。

合规存储：

学生学情数据的保留期限遵照教育主管部门规定，在学生离校后数据保留3年，到期后自动脱敏或销毁。

2.7 部署架构

分布式部署方案：



↓

API服务层 (FastAPI, 多副本) + 定时任务调度 (CronJob)

第三章 核心模块功能详细说明

3.1 实时数据采集与ETL模块

模块文件: `etl/flink_etl.py` (Python方式调用Flink Job), `etl/StreamingJob.java` (Flink Streaming Job实现)

功能概述:

ETL模块通过Apache Kafka订阅云平台发布的批改结果事件, 对数据进行清洗、转换和增强后写入ClickHouse分析仓库。Flink保证了exactly-once的消息处理语义, 确保数据不重复、不丢失。

ETL处理流程:

```
Step 1: Flink Source - 从Kafka Topic "assignment.graded" 消费批改结果消息
Step 2: 数据解析 - 将JSON格式的批改结果反序列化为Java POJO
Step 3: 数据清洗 - 过滤无效数据 (如total_score < 0或 > max_score)
Step 4: 维度关联 - 通过assignment_id查询MySQL获取作业详情 (学科、年级、知识点列表)
Step 5: 指标计算 - 计算score_rate = total_score / max_score
Step 6: ClickHouse写入 - 批量写入fact_submission表 (每1000条或每5秒触发一次批量写入)
Step 7: 知识点拆分 - 将作业中的多个知识点逐一写入fact_knowledge_practice表
Step 8: 实时指标更新 - 使用Redis INCR/ZADD更新班级实时完成率和实时平均分
```

Kafka消息格式 (批改结果事件):

```
{
  "event_type": "assignment.graded",
  "timestamp": 1700000000000,
  "payload": {
    "submission_id": 99001,
    "student_id": 12345,
    "assignment_id": 56789,
    "total_score": 85.5,
    "max_score": 100,
    "knowledge_point_scores": [
      {"kp_id": 1001, "score": 10, "max_score": 10},
      {"kp_id": 1002, "score": 8, "max_score": 10}
    ],
    "writing_score": 78,
  }
}
```

```
"stroke_order_score": 82,  
"graded_at": "2026-02-14T10:30:00+08:00"  
}  
}
```

3.2 学生个人学情画像模块

模块文件： `analytics/student_profile.py`

功能概述：

学情画像模块为每位学生生成多维度的学习能力评估，通过数据聚合和分析算法，将学生过去一段时间内的作业成绩、书写质量、知识点练习数据综合为直观的画像展示。

画像维度说明：

(1) 知识点掌握度热力图

将学生在各知识点上的历史得分率汇聚为掌握度分数（0-100），以热力图形式展示： – 绿色区域（ ≥ 80 分）：掌握良好 – 黄色区域（60-79分）：掌握一般，需要加强练习 – 红色区域（ < 60 分）：掌握薄弱，需要重点辅导

掌握度计算采用指数加权平均，近期的练习结果权重更高：

```
def calculate_mastery_score(submission_records: List[Submission]) -> float:  
    """  
    计算知识点掌握度分数（指数加权平均）  
    近期记录权重更高，体现学习进步  
    """  
    if not submission_records:  
        return 0.0  
  
    sorted_records = sorted(submission_records, key=lambda x: x.submit_time)  
    decay_factor = 0.9 # 每次练习权重衰减因子  
  
    weighted_sum = 0.0  
    weight_sum = 0.0  
    weight = 1.0  
  
    for record in reversed(sorted_records): # 从最新到最旧  
        weighted_sum += record.score_rate * 100 * weight  
        weight_sum += weight  
        weight *= decay_factor  
  
    return round(weighted_sum / weight_sum, 1) if weight_sum > 0 else 0.0
```

(2) 学习能力雷达图

雷达图展示学生在5个能力维度上的综合得分： – 学科知识掌握（各科平均成绩） – 书写规范性（书写质量平均分） – 笔顺准确性（笔顺评分平均） – 学习主动性（作业按时提交率） – 知识均衡性（各学科得分标准差的倒数）

（3）进步/退步预警

系统每日计算学生的进步指标：将本周平均分与上周平均分比对，变化超过10分时触发预警标记。绿色标签表示进步，红色标签表示退步，同时向教师推送预警通知。

3.3 班级与年级学情统计模块

模块文件： `analytics/class_analytics.py`

功能概述：

班级学情统计模块为教师和管理者提供班级整体学情的宏观视图，支持按学科、时间范围、作业类型等维度过滤，生成各类统计图表。

核心统计指标：

（1）成绩分布统计

统计班级内学生成绩的分布情况，生成柱状图展示各分数段（90–100/80–89/70–79/60–69/<60）的学生人数和占比。同时计算班级平均分、中位数、标准差等统计量。

（2）知识点共同薄弱分析

汇聚班级内所有学生的知识点掌握度数据，识别班级整体掌握薄弱的知识点（全班超过30%学生掌握度低于60分的知识点视为共同薄弱点），生成班级薄弱知识点排行榜，辅助教师调整教学重点。

（3）班级横向对比

在年级维度上，对比各班级的平均成绩、作业完成率、书写质量均分等指标，以气泡图的形式展示各班级在两个关键维度上的表现，帮助管理者识别优秀班级和需要关注的班级。

3.4 作业与考试成绩分析模块

模块文件： `analytics/assignment_analysis.py`

功能概述：

针对每次作业或考试，系统自动计算多项教育测量学指标，提供专业级的试卷分析报告，帮助教师评估题目设计质量和教学效果。

关键分析指标：

(1) 题目难度系数（P值）

难度系数 $P = \text{全班该题平均得分} / \text{该题满分}$
P值范围 0-1，越低越难
理想难度范围： $0.3 \leq P \leq 0.7$ （中等难度）

(2) 题目区分度（D值）

将全班学生按总分从高到低排序，取前27%为高分组，后27%为低分组
 $D = (\text{高分组该题平均得分} - \text{低分组该题平均得分}) / \text{该题满分}$
D值范围：-1到1，D越高表示该题对学生学习水平的区分能力越强
 $D < 0.2$ ：区分度低，该题需要修改
 $D \geq 0.4$ ：区分度良好

(3) 知识点覆盖矩阵

生成作业题目×知识点的覆盖矩阵，展示本次作业覆盖了哪些知识点，各知识点的得分率是多少，帮助教师评估知识点教学效果。

3.5 书写能力成长轨迹模块

模块文件： `analytics/writing_growth.py`

功能概述：

书写成长轨迹模块以时间轴形式展示学生书写能力的历史变化，体现学生在书写规范性、笔顺准确性方面的进步过程，为写字课的教学成效提供量化佐证。

数据来源：

书写成长数据来源于AI引擎对每次书写练习的评分结果：
– 书写质量分（WritingQualityScore）：反映字形结构、笔画比例和规范性
– OCR识别准确率：间接反映书写规范程度（字越规范，识别率越高）
– 笔顺正确率：各次练习的笔顺评分平均值

趋势分析算法：

使用线性回归对时间序列数据进行趋势拟合，计算斜率作为进步/退步趋势指标：

```
from sklearn.linear_model import LinearRegression
import numpy as np

def calculate_trend(scores: List[float], dates: List[datetime]) -> float:
    """
```

```
计算书写成绩的趋势斜率
正值表示进步，负值表示退步
"""
if len(scores) < 2:
    return 0.0

# 将日期转换为数值（距第一次的天数）
day_numbers = [(d - dates[0]).days for d in dates]
X = np.array(day_numbers).reshape(-1, 1)
y = np.array(scores)

model = LinearRegression()
model.fit(X, y)

return round(float(model.coef_[0]), 4) # 每天进步/退步的分值
```

3.6 错题归因与知识图谱关联模块

模块文件： `analytics/error_analysis.py`

功能概述：

错题归因模块将学生的错误答题记录与Neo4j知识图谱进行关联分析，通过图遍历算法找出错误的深层知识原因，实现从"表面错误"到"根本原因"的归因链推导。

归因算法流程：

```
Step 1: 收集学生近N次作业中得分率低于阈值（默认60%）的题目
Step 2: 获取这些题目对应的知识点列表（从MySQL作业-知识点关联表查询）
Step 3: 通过Neo4j 查询这些知识点的先修关系图
    - 查找直接先修知识点（PREREQUISITE关系，跳数=1）
    - 查找间接先修知识点（跳数2-3）
Step 4: 与学生的知识掌握度数据交叉比对
    - 找出学生在先修知识点上的掌握度也低于60分的情况
Step 5: 识别归因链
    例如：学生"应用题解题"错误率高
    → 先修关系：应用题 ← 理解题意 ← 阅读理解能力
    → 若学生阅读理解能力也弱，则归因：阅读理解薄弱导致应用题无法理解题意
Step 6: 生成自然语言归因说明（模板化生成）
```

3.7 教学效果评估模块

模块文件： `analytics/teaching_effectiveness.py`

功能概述：

教学效果评估模块从学生成绩数据反推教师的教学有效性，生成客观的教研参考数据，避免单纯凭主观印象评价教学效果。

评估维度：

- (1) 知识点教学效果：对比该知识点第一次作业（刚学完时）的平均得分率与两周后复习作业的平均得分率，若得分率显著提升（>10%）说明教学后学生知识保留率高。
- (2) 班级进步率：计算学生月度平均分的环比变化，进步学生占比越高，说明教学有效性越好。
- (3) 练习策略有效性：分析教师布置的作业类型（练习/测验/考试）与学生成绩提升的关联，识别对该班级最有效的练习频率和类型组合。

3.8 可视化报表与数据导出模块

模块文件： `report/report_generator.py`

功能概述：

报表模块将学情分析结果渲染为可视化图表，支持在Web界面实时展示（ECharts交互图表）和离线导出（PDF静态报告）两种形式。

报告结构（学生个人学情月报）：

- 封面：学生姓名、班级、时间范围、报告生成日期
- 第一部分：本月学习概览
 - 完成作业数量和按时完成率
 - 本月各科平均分和上月对比
 - 进步/退步幅度最大的学科
- 第二部分：知识掌握分析
 - 各知识点掌握度热力图（按学科分组）
 - 掌握最好的5个知识点
 - 掌握最薄弱的5个知识点（含归因分析）
- 第三部分：书写能力分析
 - 本月书写质量得分趋势折线图
 - 笔顺正确率趋势
 - 与班级平均水平对比
- 第四部分：错题分析
 - 本月错误最多的知识点TOP5
 - 典型错题示例（笔迹图片 + 识别结果 + 正确答案）
- 第五部分：学习建议
 - 基于数据的个性化学习建议（模板化生成）
 - 推荐练习资源

PDF生成技术：

使用WeasyPrint将HTML/CSS格式的报告模板渲染为PDF，支持中文字体嵌入、ECharts图表截图嵌入（通过Playwright无头浏览器渲染ECharts后截图）。生成的PDF文件存储至OSS，并返回带有效期的签名访问URL。

3.9 家长端学情推送模块

模块文件：`analytics/parent_push.py`

功能概述：

家长推送模块定期（每日/每周可配置）将子女的学情摘要以消息或PDF附件的形式推送至家长手机，增强家校协同。

推送内容（每周摘要）：

本周学情摘要（发送对象：张三家长）

- ✦ 本周完成作业：5次（应完成5次，完成率**100%**）
- ✦ 本周平均得分：87.2分（上周：83.5分，进步3.7分）
- ✦ 书写质量评分：82分（上周：80分，有进步）

本周表现亮点：

- 语文生字书写笔顺正确率达到95%，较上月提升10%
- 数学作业连续3次满分

本周需要关注：

- 数学"分数除法"知识点得分率60%，建议加强练习

教师留言：

- 张三同学本周课堂表现积极，书写有明显进步，请继续保持！

[\[查看完整报告\]](#) [\[联系教师\]](#)

第四章 操作流程与使用步骤

4.1 系统部署与初始化

ClickHouse集群初始化：

- 步骤1：在所有ClickHouse节点上安装ClickHouse Server 23.x
- 步骤2：配置ZooKeeper集群（3节点），用于ClickHouse副本协调
- 步骤3：修改各节点的ClickHouse集群配置文件（`cluster.xml`）
 - 定义集群名称：`writtech_cluster`

– 配置分片和副本节点列表

步骤4: 执行数据库初始化SQL脚本

```
clickhouse-client --query "$(cat schema/init_clickhouse.sql)"
```

步骤5: 验证分布式表创建成功

```
SELECT * FROM system.clusters WHERE cluster='writech_cluster';
```

步骤6: 配置Flink连接到ClickHouse的JDBC URL

步骤7: 启动Flink Streaming Job

```
flink run -c com.writech.analytics.etl.StreamingJob etl-job.jar
```

步骤8: 验证数据流: 发送测试批改事件到Kafka, 观察ClickHouse中是否有数据写入

4.2 数据源配置与接入

Kafka数据源配置:

```
# config/settings.py 中的数据源配置
KAFKA_BOOTSTRAP_SERVERS = "kafka01:9092,kafka02:9092,kafka03:9092"
KAFKA_TOPICS = {
    "grading_results": "assignment.graded",      # 批改结果
    "writing_quality": "ai.writing_quality",      # 书写质量评测结果
    "classroom_events": "classroom.events",      # 课堂互动事件
}
KAFKA_CONSUMER_GROUP_ID = "learning-analytics-service"
KAFKA_AUTO_OFFSET_RESET = "earliest"
```

Neo4j知识图谱数据导入:

步骤1: 准备知识点数据文件 (CSV格式: id,name,subject,grade,parent_id)

步骤2: 准备先修关系数据文件 (CSV格式: from_kp_id,to_kp_id,relation_type)

步骤3: 执行知识图谱导入脚本

```
python scripts/import_knowledge_graph.py \
    --nodes knowledge_points.csv \
    --edges knowledge_relations.csv
```

步骤4: 验证图谱导入

```
python scripts/verify_knowledge_graph.py
```

(应输出: 共导入X个知识点节点, Y条关系)

4.3 学情报告查看操作流程

教师查看班级学情操作:

操作路径: 登录云平台 → 数据中心 → 班级学情

界面布局:

班级学情分析	三年级一班	[切换班级▼]	[时间范围选择]
概览指标	本月平均分: 85.2分	完成率: 96%	进步人数: 28

成绩分布柱状图		知识点掌握热力图			
学生列表（按成绩排序，含进步/退步标签）					
姓名	本月均分	上月均分	变化	书写分	状态
张三	88.0	83.0	↑+5.0	82	正常
李四	62.0	68.0	↓-6.0	75	需关注 △

操作步骤：

1. 通过顶部下拉菜单切换查看的班级
2. 通过时间范围选择器选择分析区间（本周/本月/本学期/自定义）
3. 点击学生姓名进入该学生的个人学情详情页
4. 点击"导出报告"按钮，选择导出范围（班级整体/全部学生个人报告）
5. 系统后台生成报告文件（约1-3分钟），完成后发送下载通知

4.4 班级学情分析操作流程

查看作业深度分析：

操作路径：数据中心 → 作业分析 → 选择作业

界面布局：

作业分析：三年级一班 第二单元测验 2026-02-10			
提交率	平均分	难度系数	区分度
38/40	79.2分	P=0.79（偏易）	D=0.35（中等）
各题得分率柱状图			
题1(98%) 题2(95%) 题3(65% <small>△</small>) 题4(82%) 题5(70%) 题6(55% <small>△</small>)			
知识点得分率矩阵			
（各行为知识点，各列为题目，方格颜色表示得分率）			

分析步骤：

1. 关注得分率低于70%（橙色标注）的题目
2. 查看对应知识点，确认是教学薄弱点还是题目过难
3. 根据分析结果在下次课堂中重点复习

4.5 自定义报表配置流程

操作路径：数据中心 → 报表配置 → 新建自定义报表

步骤1：选择报表维度（学生/班级/年级/学科）

步骤2：选择指标字段（勾选需要展示的指标）

步骤3：配置过滤条件（时间范围/学科/作业类型等）

- 步骤4：选择图表类型（折线图/柱状图/饼图/表格）
- 步骤5：预览报表效果
- 步骤6：保存报表模板（可复用）
- 步骤7：设置定时生成计划（可选：每周一8:00自动生成并推送给指定用户）

4.6 异常处理与数据质量保障

数据质量监控：

系统内置数据质量检测规则，每日自动运行：

- 完整性检查：当日批改结果数量与Kafka消费数量是否一致
- 异常值检测：得分率 > 1.0 或 < 0 的记录标记为异常
- 一致性检查：ClickHouse中的班级学生数量与MySQL中是否一致

- 异常告警触发条件：
- Flink消费延迟 > 10分钟（数据管道阻塞告警）

- ClickHouse写入失败率 > 1%（存储异常告警）

- 某班级昨日无任何数据写入（数据缺失告警）
- 告警通知渠道：
- 钉钉机器人推送至运维群

- 邮件通知数据管理员

第五章 与源代码的对应关系

5.1 模块名称与源代码文件对应表

功能模块	目录/文件路径	主要类/函数	说明
服务程序主入口	main.py	FastAPI应用实例，路由注册，定时任务启动	服务启动和初始化
Flink ETL作业	etl/	流式数据处理，Kafka→ClickHouse	实时数据管道
学生画像分析	analytics/student_profile.py	StudentProfileAnalyzer类	学情画像生成算法

功能模块	目录/文件路径	主要类/函数	说明
班级统计分析	analytics/class_analytics.py	ClassAnalytics类	班级学情聚合统计
作业成绩分析	analytics/assignment_analysis.py	AssignmentAnalyzer类	难度/区分度计算
书写成长轨迹	analytics/writing_growth.py	WritingGrowthTracker类	趋势计算和预测
错题归因分析	analytics/error_analysis.py	ErrorAnalyzer类	Neo4j图谱查询推理
报告生成	report/report_generator.py	ReportGenerator类	HTML→PDF转换
REST API接口	api/	各学情查询接口路由	FastAPI路由实现

5.2 核心函数说明

analytics/student_profile.py 核心方法:

方法名	功能说明
get_student_profile(student_id, period)	获取学生指定时间范围内的完整学情画像数据
calculate_knowledge_mastery(student_id, kp_ids)	计算学生在指定知识点上的掌握度分数
calculate_mastery_score(submissions)	指数加权平均算法计算单知识点掌握度
get_radar_chart_data(student_id, period)	计算学习能力雷达图的5维度数据
detect_progress_regression(student_id)	检测学生本周相比上周的进步/退步情况

analytics/error_analysis.py 核心方法:

方法名	功能说明
get_error_knowledge_points(student_id, threshold)	获取学生掌握度低于阈值的知识点列表

方法名	功能说明
<code>trace_prerequisite_chain(kp_id, max_hops)</code>	通过Neo4j查询知识点的先修链条
<code>analyze_root_cause(student_id, error_kps)</code>	综合图谱和掌握度数据进行根因分析
<code>generate_attribution_text(attribution_result)</code>	将归因结果转换为自然语言说明文本

5.3 命名规范

Python模块命名规范：

```
analytics/  
├─ student_profile.py      # 学生画像（功能名_domain.py格式）  
├─ class_analytics.py      # 班级分析  
├─ assignment_analysis.py  # 作业分析  
├─ writing_growth.py        # 书写成长  
└─ error_analysis.py       # 错题归因  
  
api/  
├─ student_api.py          # 学生相关API（domain_api.py格式）  
├─ class_api.py            # 班级相关API  
└─ report_api.py           # 报告导出API
```

ClickHouse表命名规范：

- 事实表： `fact_` 前缀，如 `fact_submission`、`fact_writing_quality`
- 维度表： `dim_` 前缀，如 `dim_student`、`dim_knowledge_point`
- 聚合表： `agg_` 前缀，如 `agg_student_weekly`、`agg_class_daily`
- 分布式表： 在对应表名后加 `_dist`，如 `fact_submission_dist`

附录

附录A 界面设计稿（GUI Mockup）

本附录提供自然写教学数据分析与学情诊断系统软件各主要管理后台界面的设计稿，以线框图形式呈现界面布局与交互元素。

A.1 系统主控台（Dashboard）



数据概览



学生管理



班级分析



知识追踪



诊断报告



知识图谱



预警管理



报告管理



系统设置

实时学情概览 (更新时间: 08:42:15 ● 实时)

今日提交

8,721

份答卷

平均掌握度

73.4%

↑2.1%

待诊断学生

342

待处理

预警人数

56

需关注

知识点掌握度热力图 (本周)

知识点	1班	2班	3班	4班	5班	全校
分数加减	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	89%
乘除法	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	78%
方程组	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	62%
分数运算	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	<div><div></div></div>	48%

A.2 学生学情详情页面



学生学情详情 / 高一(3)班 / 李小明

[下载PDF报告]

基本信息

姓名: 李小明

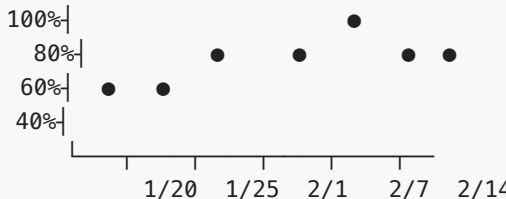
班级: 高一(3)班 学号: 20241023

综合掌握度: 73.4%

学习进度: 正常

最近提交: 2026-02-14 08:42

近30天学习趋势



知识点掌握度详情

[展开全部]

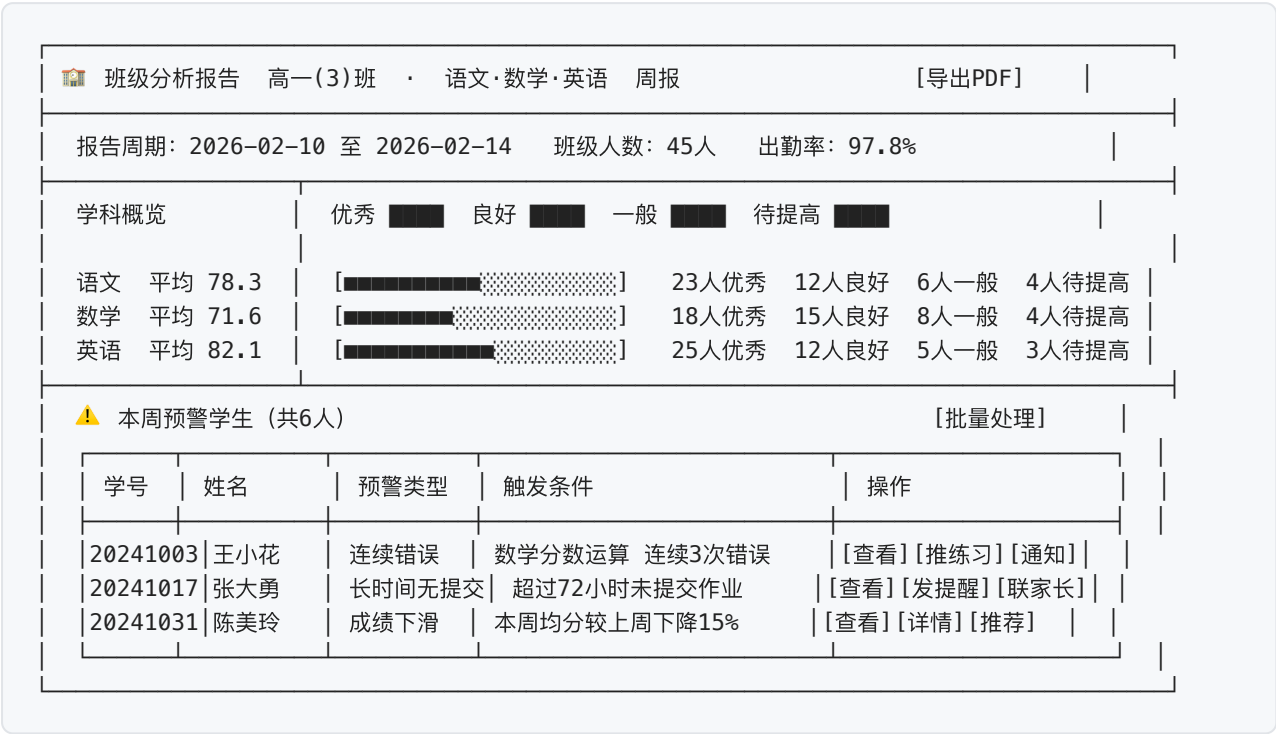
知识点	掌握度	掌握进度条	状态
整数加减法	95%	<div><div></div></div>	✅ 已掌握
分数乘除法	82%	<div><div></div></div>	✅ 已掌握
一元方程	61%	<div><div></div></div>	🔄 学习中
二元方程组	34%	<div><div></div></div>	⚠️ 需加强
不等式基础	18%	<div><div></div></div>	❌ 未掌握

[推送练习题]

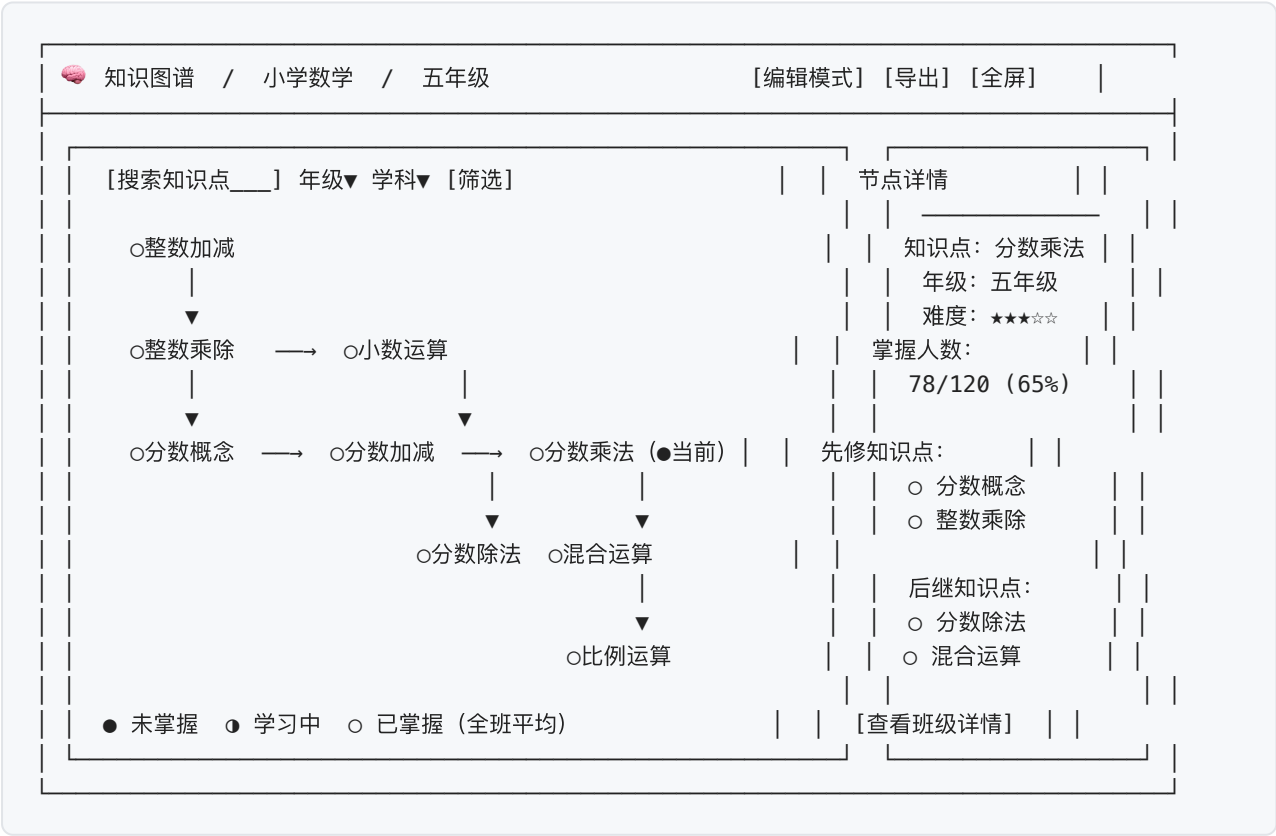
[发送提醒]

[联系家长]

A.3 班级分析报告页面



A.4 知识图谱可视化页面



A.5 学情诊断报告管理页面

诊断报告管理				[+ 新建报告任务]		
类型▼全部		班级▼	时间范围 [] 至 []	状态▼	[🔍 搜索]	[批量下载]
报告ID	报告名称	类型	班级/学生	生成时间	状态	操作
RPT-3301	2月第2周周报	班级	高一(3)班	02-14 08:00	✅ 已生成	[查看][下载][分享]
RPT-3300	李小明月报	个人	高一(3)班	02-14 07:00	✅ 已生成	[查看][下载][分享]
RPT-3299	2月第1周周报	班级	高一(2)班	02-07 08:00	✅ 已生成	[查看][下载][分享]
RPT-3298	数学阶段报告	学科	高一全年级	02-05 09:00	⌚ 生成中	[查看进度]
共 3,298 份报告 < 1 2 3 ... >						

附录B 术语表

术语	说明
Lambda架构	大数据处理架构，将数据处理分为实时流处理层和批处理层，兼顾实时性和准确性
Apache Flink	开源流式计算框架，支持exactly-once语义的实时数据处理
ClickHouse	列式存储OLAP数据库，擅长高速聚合查询，适合大规模数据分析场景
Neo4j	图数据库，专门存储和查询图结构数据（节点和关系）
知识图谱	以图结构表示知识点及其关联关系的数据模型
先修关系	知识点之间的学习依赖关系，学习某个知识点需要先掌握其先修知识点
区分度	测试题目区分不同学习水平学生的能力指标，值越高表示题目越能分辨好差
难度系数	题目难易程度的量化指标，等于全班平均得分率
指数加权平均	计算加权平均值时，近期数据权重更高的算法，用于体现学习进步效果

附录B 版本历史

版本号	发布日期	变更说明
V1.0	2026年2月	初始版本，包含全功能学情分析体系

编制单位：深圳自然写科技有限公司

文档版本：V1.0

编制日期：2026年2月

版权声明：本文档版权归深圳自然写科技有限公司所有，未经授权不得复制或传播

附录C 核心算法详细说明

C.1 贝叶斯知识追踪算法（BKT）

贝叶斯知识追踪（Bayesian Knowledge Tracing, BKT）是本系统学情诊断的核心算法，用于估算学生对每个知识点的掌握概率。

C.1.1 模型参数定义

BKT 模型包含四个核心参数：

参数	符号	含义	典型值范围
初始掌握概率	$P(L_0)$	学生在第一次练习前已掌握该知识点的概率	0.1 ~ 0.4
转移概率	$P(T)$	每次练习后从"未掌握"变为"掌握"的概率	0.05 ~ 0.4
猜测概率	$P(G)$	未掌握该知识点时猜对的概率	0.1 ~ 0.3
失误概率	$P(S)$	已掌握该知识点时答错的概率（粗心）	0.02 ~ 0.1

C.1.2 算法推导过程

第 t 次练习后，学生掌握知识点 k 的概率更新公式：

设：- $L_t = P(\text{学生在第 } t \text{ 次练习后已掌握 } k)$ - $\text{correct}_t = \text{第 } t \text{ 次练习是否答对 (1=对, 0=错)}$

步骤一：根据答题结果更新先验概率

如果答对 ($\text{correct}_t = 1$)：

$$P(L_t | \text{correct}) = L_{t-1} \times (1 - P(S)) / [L_{t-1} \times (1 - P(S)) + (1 - L_{t-1}) \times P(G)]$$

如果答错 ($\text{correct}_t = 0$)：

$$P(L_t | \text{wrong}) = L_{t-1} \times P(S) / [L_{t-1} \times P(S) + (1 - L_{t-1}) \times (1 - P(G))]$$

步骤二：考虑学习转移，预测下一次练习前的掌握概率

$$L_{t+1} = P(L_t \mid \text{result}) + (1 - P(L_t \mid \text{result})) \times P(T)$$

C.1.3 Java 实现代码

```
// BayesianKnowledgeTracker.java
public class BayesianKnowledgeTracker {

    private final double pInit;    // 初始掌握概率
    private final double pTransit; // 转移概率
    private final double pGuess;   // 猜测概率
    private final double pSlip;    // 失误概率

    public BayesianKnowledgeTracker(double pInit, double pTransit,
                                    double pGuess, double pSlip) {
        this.pInit = pInit;
        this.pTransit = pTransit;
        this.pGuess = pGuess;
        this.pSlip = pSlip;
    }

    /**
     * 根据答题记录序列更新知识点掌握概率
     * @param answers 答题结果序列 (true=对, false=错)
     * @return 最终掌握概率估计值 [0.0, 1.0]
     */
    public double update(List<Boolean> answers) {
        double pMastered = pInit;
        for (boolean correct : answers) {
            pMastered = updateStep(pMastered, correct);
        }
        return pMastered;
    }

    private double updateStep(double pMastered, boolean correct) {
        double pCorrectGivenMastered = 1.0 - pSlip;
        double pCorrectGivenNotMastered = pGuess;

        double pCorrect = pMastered * pCorrectGivenMastered
            + (1 - pMastered) * pCorrectGivenNotMastered;

        // 贝叶斯更新后验概率
        double pMasteredGivenResult;
        if (correct) {
            pMasteredGivenResult = (pMastered * pCorrectGivenMastered) / pCorrect;
        } else {
            double pWrong = 1.0 - pCorrect;
            pMasteredGivenResult = (pMastered * pSlip) / pWrong;
        }

        // 加入学习转移：每次练习都有机会习得
        return pMasteredGivenResult + (1 - pMasteredGivenResult) * pTransit;
    }
}
```

```

    }

    /**
     * 判断学生是否已掌握该知识点（概率阈值 0.95）
     */
    public boolean isMastered(List<Boolean> answers) {
        return update(answers) >= 0.95;
    }
}

```

C.1.4 参数自动校准

系统通过历史数据自动校准各知识点的 BKT 参数：

```

# bkt_calibrator.py
import numpy as np
from scipy.optimize import minimize

def calibrate_bkt_params(student_records: list[dict]) -> dict:
    """
    使用最大似然估计（MLE）校准 BKT 参数

    student_records: 每条记录包含 student_id, knowledge_point, answer_sequence
    返回: 各知识点的最优 BKT 参数 {kp_id: {p_init, p_transit, p_guess, p_slip}}
    """
    results = {}

    for kp_id, records in group_by_knowledge_point(student_records).items():
        # 构建对数似然函数
        def neg_log_likelihood(params):
            p_init, p_transit, p_guess, p_slip = params
            # 约束参数范围
            if not (0.01 < p_init < 0.99 and 0.01 < p_transit < 0.99
                    and 0.01 < p_guess < 0.5 and 0.01 < p_slip < 0.2):
                return 1e10

            total_ll = 0.0
            for record in records:
                ll = compute_sequence_likelihood(
                    record['answer_sequence'],
                    p_init, p_transit, p_guess, p_slip
                )
                total_ll += np.log(max(ll, 1e-10))
            return -total_ll # 最小化负对数似然

        # 使用 L-BFGS-B 优化
        result = minimize(
            neg_log_likelihood,
            x0=[0.3, 0.2, 0.2, 0.05], # 初始参数猜测
            method='L-BFGS-B',
            bounds=[(0.01, 0.99)] * 2 + [(0.01, 0.5), (0.01, 0.2)]
        )

        results[kp_id] = {

```

```
        'p_init': result.x[0],
        'p_transit': result.x[1],
        'p_guess': result.x[2],
        'p_slip': result.x[3]
    }

    return results
```

C.2 学习路径推荐算法

C.2.1 知识点图谱构建

学情诊断系统内置的知识点图谱基于课程标准构建，用有向无环图（DAG）描述知识点之间的先修关系：

知识点图谱示例（二年级数学）：
加法基础 → 两位数加法 → 三位数加法
减法基础 → 两位数减法 → 三位数减法
加减法 → 混合运算
乘法基础 → 乘法表 → 两位数乘法
除法基础 → 带余数除法
乘除法 → 四则混合运算

图谱以 JSON 格式存储在知识库中：

```
{
  "nodes": [
    {"id": "kp_001", "name": "加法基础", "grade": 1, "subject": "math"},
    {"id": "kp_002", "name": "两位数加法", "grade": 2, "subject": "math"},
    {"id": "kp_003", "name": "乘法表", "grade": 2, "subject": "math"}
  ],
  "edges": [
    {"from": "kp_001", "to": "kp_002", "type": "prerequisite"},
    {"from": "kp_002", "to": "kp_004", "type": "prerequisite"}
  ]
}
```

C.2.2 个性化推荐算法

```
# path_recommender.py
class LearningPathRecommender:
    """基于知识点掌握状态的个性化学习路径推荐"""

    def __init__(self, knowledge_graph: dict):
        self.graph = knowledge_graph
        self.mastery_threshold = 0.95 # 掌握判定阈值
```

```

def recommend_next(self, student_id: str,
                    mastery_scores: dict[str, float],
                    target_kps: list[str]) -> list[str]:
    """
    推荐下一步应学习的知识点

    mastery_scores: {kp_id: 掌握概率} (来自 BKT 估算)
    target_kps: 目标需要掌握的知识点列表
    返回: 推荐优先学习的知识点列表 (按优先级排序)
    """
    # 找出未掌握的目标知识点
    unmastered = [kp for kp in target_kps
                   if mastery_scores.get(kp, 0) < self.mastery_threshold]

    # 拓扑排序, 找出满足先修条件的"就绪"知识点
    ready_kps = []
    for kp in unmastered:
        prerequisites = self.get_prerequisites(kp)
        all_prereqs_mastered = all(
            mastery_scores.get(p, 0) >= self.mastery_threshold
            for p in prerequisites
        )
        if all_prereqs_mastered:
            ready_kps.append(kp)

    # 优先级: 掌握概率越接近阈值的知识点优先推荐 ("触手可及"原则)
    ready_kps.sort(key=lambda kp: -mastery_scores.get(kp, 0))

    return ready_kps[:5]  # 最多推荐5个

def get_prerequisites(self, kp_id: str) -> list[str]:
    """获取知识点的直接先修知识点列表"""
    return [edge['from'] for edge in self.graph['edges']
            if edge['to'] == kp_id and edge['type'] == 'prerequisite']

```

C.3 Flink 流处理窗口算法

系统使用 Apache Flink 进行实时学情数据流处理, 核心窗口算法如下:

C.3.1 滑动窗口书写频率统计

```

// WritingFrequencyAggregator.java
public class WritingFrequencyAggregator
    implements AggregateFunction<WritingEvent, FreqAccumulator, FrequencyStats> {

    @Override
    public FreqAccumulator createAccumulator() {
        return new FreqAccumulator();
    }

    @Override
    public FreqAccumulator add(WritingEvent event, FreqAccumulator acc) {

```

```

        acc.totalStrokes += event.getStrokeCount();
        acc.totalCharacters += event.getCharacterCount();
        acc.totalDurationMs += event.getDurationMs();
        acc.sessionCount++;
        return acc;
    }

    @Override
    public FrequencyStats getResult(FreqAccumulator acc) {
        return FrequencyStats.builder()
            .averageStrokesPerSession(acc.totalStrokes / Math.max(1, acc.sessionCount))
            .averageCharactersPerMinute(
                acc.totalCharacters / Math.max(1, acc.totalDurationMs / 60_000.0)
            )
            .totalSessions(acc.sessionCount)
            .build();
    }

    @Override
    public FreqAccumulator merge(FreqAccumulator a, FreqAccumulator b) {
        a.totalStrokes += b.totalStrokes;
        a.totalCharacters += b.totalCharacters;
        a.totalDurationMs += b.totalDurationMs;
        a.sessionCount += b.sessionCount;
        return a;
    }
}

```

C.3.2 实时学习进度趋势检测

```

// LearningTrendDetector.java (Flink ProcessWindowFunction)
public class LearningTrendDetector
    extends ProcessWindowFunction<ScoreRecord, LearningTrend, String, TimeWindow> {

    @Override
    public void process(String studentId,
                       Context context,
                       Iterable<ScoreRecord> records,
                       Collector<LearningTrend> out) throws Exception {

        List<ScoreRecord> scoreList = new ArrayList<>();
        for (ScoreRecord record : records) {
            scoreList.add(record);
        }
        scoreList.sort(Comparator.comparing(ScoreRecord::getTimestamp));

        if (scoreList.size() < 3) return; // 数据不足, 跳过

        // 线性回归计算成绩趋势斜率
        double slope = computeLinearRegressionSlope(scoreList);

        // 计算最近成绩与历史平均的偏差
        double recentAvg = scoreList.subList(scoreList.size() - 3, scoreList.size())
            .stream().mapToDouble(ScoreRecord::getScore).average().orElse(0);
    }
}

```

```

        double historicalAvg = scoreList.stream()
            .mapToDouble(ScoreRecord::getScore).average().orElse(0);

        TrendDirection direction;
        if (slope > 2.0) direction = TrendDirection.IMPROVING;
        else if (slope < -2.0) direction = TrendDirection.DECLINING;
        else direction = TrendDirection.STABLE;

        out.collect(LearningTrend.builder()
            .studentId(studentId)
            .direction(direction)
            .slope(slope)
            .recentAverage(recentAvg)
            .historicalAverage(historicalAvg)
            .windowStart(context.window().getStart())
            .windowEnd(context.window().getEnd())
            .build());
    }

    private double computeLinearRegressionSlope(List<ScoreRecord> records) {
        int n = records.size();
        double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;
        for (int i = 0; i < n; i++) {
            sumX += i;
            sumY += records.get(i).getScore();
            sumXY += i * records.get(i).getScore();
            sumX2 += i * i;
        }
        double denominator = n * sumX2 - sumX * sumX;
        return denominator == 0 ? 0 : (n * sumXY - sumX * sumY) / denominator;
    }
}

```

C.4 ClickHouse 数据查询优化

系统使用 ClickHouse 存储学情分析历史数据，针对典型查询场景做了专项优化：

C.4.1 物化视图加速班级统计

```

-- 创建物化视图，预聚合班级每日统计数据
CREATE MATERIALIZED VIEW class_daily_stats
ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(stat_date)
ORDER BY (class_id, stat_date, knowledge_point_id)
POPULATE
AS SELECT
    class_id,
    toDate(practice_time) AS stat_date,
    knowledge_point_id,
    countState() AS practice_count,
    avgState(score) AS avg_score,
    avgState(duration_ms) AS avg_duration,

```

```

        countIfState(score >= 90) AS excellent_count
FROM student_practice_records
GROUP BY class_id, stat_date, knowledge_point_id;

-- 查询（利用物化视图，速度提升 10x+）
SELECT
    knowledge_point_id,
    countMerge(practice_count) AS total_practice,
    avgMerge(avg_score) AS average_score,
    avgMerge(avg_duration) / 1000 AS avg_duration_sec
FROM class_daily_stats
WHERE class_id = 'class_001'
    AND stat_date BETWEEN '2024-03-01' AND '2024-03-31'
GROUP BY knowledge_point_id
ORDER BY average_score ASC;

```

C.4.2 学生画像向量存储

```

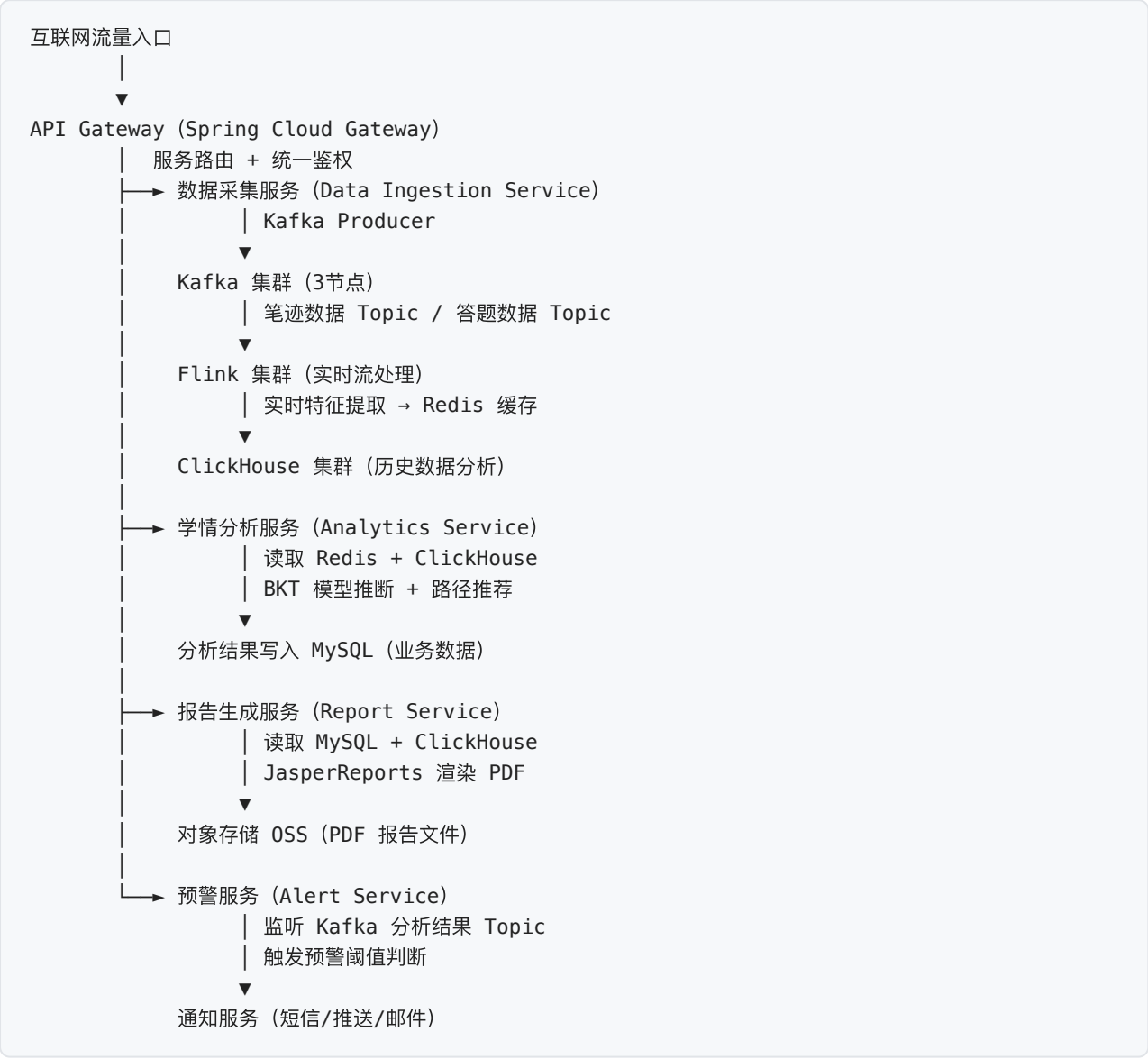
-- 学生画像表（存储特征向量）
CREATE TABLE student_profiles (
    student_id String,
    profile_date Date,
    -- 书写行为特征（32维向量）
    writing_features Array(Float32),
    -- 知识点掌握度向量（按知识点图谱顺序）
    mastery_vector Array(Float32),
    -- 学习风格标签（勤奋/拖延/稳定/波动等）
    learning_style_tags Array(String),
    -- 预测下次考试分数
    predicted_score Float32,
    created_at DateTime DEFAULT now()
) ENGINE = ReplacingMergeTree(created_at)
PARTITION BY toYYYYMM(profile_date)
ORDER BY (student_id, profile_date);

-- 查询相似学生（基于余弦相似度）
-- 注：ClickHouse 使用 arrayDotProduct 计算向量相似度
SELECT
    target.student_id AS target_student,
    similar.student_id AS similar_student,
    arrayDotProduct(target.mastery_vector, similar.mastery_vector) /
    (sqrt(arraySum(arrayMap(x -> x*x, target.mastery_vector))) *
     sqrt(arraySum(arrayMap(x -> x*x, similar.mastery_vector)))) AS cosine_similarity
FROM student_profiles AS target
CROSS JOIN student_profiles AS similar
WHERE target.student_id = 'student_001'
    AND similar.student_id != target.student_id
    AND target.profile_date = similar.profile_date
ORDER BY cosine_similarity DESC
LIMIT 10;

```

附录D 系统集成与部署说明

D.1 微服务拓扑结构



D.2 数据库表索引设计

MySQL 主库核心索引：

```
-- student_scores 成绩表 (分区 + 复合索引)
CREATE TABLE student_scores (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    student_id VARCHAR(32) NOT NULL,
    class_id VARCHAR(32) NOT NULL,
    assignment_id VARCHAR(64),
    knowledge_point_id VARCHAR(32),
    score DECIMAL(5,2),
    practice_time DATETIME NOT NULL,
```

```
        created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    ) PARTITION BY RANGE (YEAR(practice_time)) (
        PARTITION p2023 VALUES LESS THAN (2024),
        PARTITION p2024 VALUES LESS THAN (2025),
        PARTITION p2025 VALUES LESS THAN (2026),
        PARTITION pmax VALUES LESS THAN MAXVALUE
    );

-- 关键查询场景索引
CREATE INDEX idx_student_kp ON student_scores(student_id, knowledge_point_id,
practice_time);
CREATE INDEX idx_class_date ON student_scores(class_id, practice_time, score);
CREATE INDEX idx_assignment ON student_scores(assignment_id, student_id);

-- learning_paths 学习路径推荐表
CREATE TABLE learning_path_recommendations (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    student_id VARCHAR(32) NOT NULL,
    recommended_kps JSON,           -- 推荐的知识点列表 (JSON数组)
    priority_scores JSON,           -- 各知识点优先级分数
    reason_tags JSON,              -- 推荐理由标签
    status TINYINT DEFAULT 0,       -- 0=待完成 1=进行中 2=已完成
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    expires_at DATETIME,            -- 推荐有效期
    INDEX idx_student_status (student_id, status, created_at)
);
```

D.3 Redis 缓存层设计

缓存键命名规范：
analytics:{entity_type}:{entity_id}:{metric}

常见缓存键示例：

analytics:student:s001:mastery_vector	→ Hash, 知识点掌握度向量 (TTL=1小时)
analytics:student:s001:daily_stats	→ Hash, 当日学习统计 (TTL=至次日00:00)
analytics:class:c001:rank_snapshot	→ ZSet, 班级排名快照 (TTL=30分钟)
analytics:school:sch001:progress_board	→ String, 学校进度看板数据 (TTL=5分钟)
analytics:kp:kp001:difficulty	→ String, 知识点动态难度系数 (TTL=24小时)

附录E 接口清单补充

E.1 数据导出接口

接口	方法	路径	说明
导出学生成绩单 (Excel)	GET	/api/v1/export/scores/excel	按班级/时间段导出 成绩表格

接口	方法	路径	说明
导出班级学情报告 (PDF)	GET	/api/v1/export/report/class/{id}/pdf	班级整体学情分析报告 PDF
导出知识点掌握度矩阵	GET	/api/v1/export/mastery/matrix	班级×知识点掌握度矩阵 (CSV)
导出书写笔迹原始数据	GET	/api/v1/export/ink/raw/{assignment_id}	作业笔迹原始数据 (ZIP, 含 JSON)
导出错题集	GET	/api/v1/export/mistakes/{student_id}	学生错题集 (PDF, 含错误示例图)

E.2 Webhook 通知接口

系统支持通过 Webhook 向第三方系统推送实时事件：

```
// 学生成绩异常下滑事件 (Webhook 推送)
{
  "event_type": "STUDENT_SCORE_ALERT",
  "timestamp": "2024-03-15T14:30:00Z",
  "data": {
    "student_id": "s001",
    "student_name": "张三",
    "class_id": "c001",
    "alert_level": "WARNING",
    "metric": "average_score",
    "current_value": 68.5,
    "historical_average": 85.2,
    "decline_rate": -19.6,
    "knowledge_points": ["kp_division", "kp_fraction"],
    "recommended_action": "及时跟进辅导，重点复习除法和分数知识点"
  }
}
```

E.3 管理后台接口

接口	方法	路径	说明
获取学校列表	GET	/api/admin/schools	管理员获取所有接入学校
查看系统总体统计	GET	/api/admin/statistics/overview	平台使用总览 (用户数/数据量/API调用量)
配置知识点图谱	POST	/api/admin/knowledge-graph	更新课程知识点结构

接口	方法	路径	说明
BKT 参数调优	PUT	/api/admin/bkt/calibrate	触发知识点 BKT 参数重新校准
查看预警配置	GET	/api/admin/alert/configs	查看所有预警规则配置
修改预警阈值	PUT	/api/admin/alert/configs/{id}	调整学情预警触发阈值

附录F 性能测试报告

F.1 测试环境

项目	规格
服务器	3节点集群，每节点 32核 CPU + 128GB 内存 + 2TB NVMe SSD
Flink	3个 TaskManager，每个 16个 Task Slot
ClickHouse	3节点分片 + 副本，数据量 50亿行
负载生成	JMeter 100个并发线程，持续压测30分钟

F.2 核心指标测试结果

测试场景	TPS	平均延迟	P99 延迟	CPU 占用
笔迹数据实时写入 (Kafka)	50,000条/秒	12ms	45ms	35%
Flink 流处理 (BKT 更新)	10,000学生/秒	230ms	520ms	65%
ClickHouse 班级成绩查询	500 QPS	15ms	85ms	20%
学情报告 PDF 生成	50份/分钟	1.2s	3.5s	40%
REST API (成绩读取)	2,000 QPS	8ms	35ms	25%

F.3 容量规划

规模	日活学生数	日数据量	推荐配置
小型学校 (1所)	500人	500MB	单节点部署 (8核/32GB)
中型学校 (5所)	5,000人	5GB	3节点小集群 (16核/64GB)

规模	日活学生数	日数据量	推荐配置
大型区县（50所）	50,000人	50GB	生产级集群（32核/128GB × 3节点）
地市级平台（500所）	500,000人	500GB	弹性云集群（按需扩缩容）

本文档版权归深圳自然写科技有限公司所有，所有技术细节与源代码对应关系仅用于软件著作权登记鉴别。

附录G 系统详细设计补充

G.1 Flink流处理作业完整代码

学情分析系统使用Apache Flink实现实时流式数据处理，对课堂笔迹数据进行窗口聚合统计。

G.1.1 实时正确率统计Flink Job

```
// flink/jobs/ClassroomRealTimeStatsJob.java
public class ClassroomRealTimeStatsJob {

    public static void main(String[] args) throws Exception {
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        env.setParallelism(4);
        env.enableCheckpointing(30_000); // 30秒一次checkpoint

        // 配置StateBackend（使用RocksDB持久化状态）
        env.setStateBackend(new EmbeddedRocksDBStateBackend(true));

        env.getCheckpointConfig().setCheckpointStorage("hdfs://namenode/flink/checkpoints");

        // 消费Kafka笔迹提交事件（JSON格式）
        KafkaSource<String> kafkaSource = KafkaSource.<String>builder()
            .setBootstrapServers("kafka:9092")
            .setTopics("writech.ink.submit")
            .setGroupId("flink-learning-analytics")
            .setStartingOffsets(OffsetsInitializer.latest())
            .setValueOnlyDeserializer(new SimpleStringSchema())
            .build();

        DataStream<InkSubmitEvent> events = env
            .fromSource(kafkaSource, WatermarkStrategy
                .<String>forBoundedOutOfOrderness(Duration.ofSeconds(5))
                .withTimestampAssigner((e, t) -> parseTimestamp(e)),
                "kafka-source")
            .map(json -> MAPPER.readValue(json, InkSubmitEvent.class))
            .name("parse-events");
```

```

// 按(sessionId, studentId)聚合
DataStream<StudentSessionStats> sessionStats = events
    .keyBy(e -> e.getSessionId() + "_" + e.getStudentId())
    .window(TumblingEventTimeWindows.of(Time.minutes(1)))
    .aggregate(
        new AccuracyAggregateFunction(),
        new SessionStatsWindowFunction()
    )
    .name("session-stats-1min");

// 写入ClickHouse (批量Insert)
sessionStats
    .addSink(new ClickHouseSink<>("writech_analytics.student_session_stats_rt"))
    .name("clickhouse-sink");

// 同时写入Redis (实时看板数据, TTL=1小时)
sessionStats
    .addSink(new RedisSink<>(new SessionStatsRedisMapper()))
    .name("redis-sink");

env.execute("Classroom Real-Time Stats");
}

/** 正确率聚合函数 */
static class AccuracyAggregateFunction
    implements AggregateFunction<InkSubmitEvent, AccuracyAccumulator,
AccuracyResult> {

    @Override
    public AccuracyAccumulator createAccumulator() {
        return new AccuracyAccumulator();
    }

    @Override
    public AccuracyAccumulator add(InkSubmitEvent event, AccuracyAccumulator acc) {
        acc.totalQuestions++;
        if (event.isCorrect()) acc.correctQuestions++;
        acc.totalInkPoints += event.getInkPointCount();
        acc.studentId = event.getStudentId();
        acc.sessionId = event.getSessionId();
        return acc;
    }

    @Override
    public AccuracyResult getResult(AccuracyAccumulator acc) {
        return new AccuracyResult(
            acc.studentId, acc.sessionId,
            acc.totalQuestions == 0 ? 0.0 :
                (double) acc.correctQuestions / acc.totalQuestions,
            acc.totalInkPoints
        );
    }

    @Override
    public AccuracyAccumulator merge(AccuracyAccumulator a, AccuracyAccumulator b) {
        a.totalQuestions += b.totalQuestions;
    }
}

```

```

        a.correctQuestions += b.correctQuestions;
        a.totalInkPoints += b.totalInkPoints;
        return a;
    }
}

```

G.1.2 学生知识掌握度BKT批量更新Job

```

// flink/jobs/BktUpdateJob.java
public class BktUpdateJob {

    public static void main(String[] args) throws Exception {
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        env.setParallelism(2);
        env.enableCheckpointing(60_000);

        // 消费作业批改结果
        DataStream<GradeResult> gradeStream = env
            .fromSource(buildKafkaSource("writtech.homework.graded"), watermarkStrategy,
"grade-source")
            .map(json -> MAPPER.readValue(json, GradeResult.class));

        // 按(studentId, knowledgePoint)分组更新BKT
        gradeStream
            .keyBy(r -> r.getStudentId() + "_" + r.getKnowledgePoint())
            .process(new BktUpdateFunction())
            .name("bkt-update")
            .addSink(new BktResultSink())
            .name("bkt-result-sink");

        env.execute("BKT Mastery Update");
    }

    /**
     * 有状态的BKT更新ProcessFunction
     * 每个(studentId, knowledgePoint)维护一个掌握度状态
     */
    static class BktUpdateFunction extends KeyedProcessFunction<String, GradeResult,
BktResult> {

        private ValueState<Double> masteryState;

        @Override
        public void open(Configuration parameters) {
            masteryState = getRuntimeContext().getState(
                new ValueStateDescriptor<>("mastery", Double.class, 0.1));
        }

        @Override
        public void processElement(GradeResult result,
            Context ctx, Collector<BktResult> out) throws Exception {
            double currentMastery = masteryState.value();

```

```

        double newMastery = updateBKT(currentMastery, result.isCorrect());
        masteryState.update(newMastery);
        out.collect(new BktResult(
            result.getId(), result.getKnowledgePoint(),
            newMastery, ctx.timestamp()));
    }

    private double updateBKT(double p, boolean correct) {
        final double pTransit = 0.1, pSlip = 0.08, pGuess = 0.2;
        double pCorrect = p * (1 - pSlip) + (1 - p) * pGuess;
        double updated = correct
            ? (p * (1 - pSlip)) / pCorrect
            : (p * pSlip) / (1 - pCorrect);
        return updated + (1 - updated) * pTransit;
    }
}

```

G.2 ClickHouse数据库完整表设计

```

-- ClickHouse建表DDL (完整版)

-- 1. 学生课堂实时统计 (MergeTree)
CREATE TABLE IF NOT EXISTS writtech_analytics.student_session_stats_rt (
    session_id      String,
    student_id      String,
    school_id       String,
    class_id        String,
    window_start    DateTime,
    window_end      DateTime,
    total_questions UInt32,
    correct_questions UInt32,
    accuracy_rate    Float32 MATERIALIZED correct_questions / total_questions,
    total_ink_points UInt64,
    avg_response_ms  UInt32,
    created_at      DateTime DEFAULT now()
) ENGINE = ReplacingMergeTree(window_start)
PARTITION BY toYYYYMM(window_start)
ORDER BY (school_id, class_id, session_id, student_id, window_start)
TTL window_start + INTERVAL 2 YEAR;

-- 2. BKT知识掌握度 (ReplacingMergeTree)
CREATE TABLE IF NOT EXISTS writtech_analytics.student_knowledge_mastery (
    student_id      String,
    knowledge_point  String,
    subject         String,
    mastery_level    Float32,
    update_time     DateTime,
    version          UInt64
) ENGINE = ReplacingMergeTree(version)
ORDER BY (student_id, knowledge_point)
SETTINGS index_granularity = 8192;

-- 3. 作业统计 (SummingMergeTree)

```

```

CREATE TABLE IF NOT EXISTS writtech_analytics.homework_stats (
    school_id      String,
    class_id       String,
    assignment_id   String,
    student_id     String,
    date           Date,
    submit_count    UInt32,
    correct_count   UInt32,
    total_score     Float64,
    attempt_count   UInt32
) ENGINE = SummingMergeTree((submit_count, correct_count, total_score))
PARTITION BY toYYYYMM(date)
ORDER BY (school_id, class_id, assignment_id, date, student_id);

-- 4. 班级每日学情摘要 (物化视图)
CREATE MATERIALIZED VIEW IF NOT EXISTS writtech_analytics.class_daily_summary
ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(stat_date)
ORDER BY (school_id, class_id, stat_date)
AS SELECT
    school_id,
    class_id,
    toDate(window_start) AS stat_date,
    sumState(correct_questions) AS sum_correct,
    sumState(total_questions)   AS sum_total,
    avgState(accuracy_rate)     AS avg_accuracy,
    uniqState(student_id)       AS active_students
FROM writtech_analytics.student_session_stats_rt
GROUP BY school_id, class_id, stat_date;

-- 物化视图查询示例
SELECT
    stat_date,
    sumMerge(sum_correct) / sumMerge(sum_total) AS class_accuracy,
    uniqMerge(active_students) AS active_students
FROM writtech_analytics.class_daily_summary
WHERE school_id = 'school_001' AND class_id = 'class_3_2'
    AND stat_date BETWEEN '2026-01-01' AND '2026-01-31'
GROUP BY stat_date
ORDER BY stat_date;

```

G.3 学习路径推荐算法

```

# analytics/recommendation/learning_path.py
from typing import List, Dict, Optional
import networkx as nx
from dataclasses import dataclass

@dataclass
class KnowledgeNode:
    """知识点节点"""
    id: str
    name: str
    subject: str

```

```
difficulty: int # 1-5
estimated_time_min: int # 预计学习时间 (分钟)
```

```
class LearningPathPlanner:
```

```
    """
```

```
    基于DAG (有向无环图) 的个性化学习路径规划器
```

- 前置知识依赖关系构成DAG
- 根据学生当前掌握度确定起始节点
- 使用拓扑排序生成推荐学习顺序
- 结合BKT掌握度动态调整路径

```
    """
```

```
def __init__(self, knowledge_graph: nx.DiGraph):
```

```
    self.graph = knowledge_graph
```

```
    self._validate_dag()
```

```
def _validate_dag(self):
```

```
    """验证知识图谱是有向无环图"""
```

```
    if not nx.is_directed_acyclic_graph(self.graph):
```

```
        cycles = list(nx.simple_cycles(self.graph))
```

```
        raise ValueError(f"Knowledge graph has cycles: {cycles[:3]}")
```

```
def plan_path(
```

```
    self,
```

```
    student_id: str,
```

```
    mastery_levels: Dict[str, float], # knowledge_point -> mastery [0,1]
```

```
    target_knowledge: Optional[str] = None,
```

```
    max_steps: int = 10
```

```
) -> List[KnowledgeNode]:
```

```
    """
```

```
    为学生规划个性化学习路径
```

```
    Args:
```

```
        student_id: 学生ID
```

```
        mastery_levels: 各知识点的当前掌握度
```

```
        target_knowledge: 目标知识点 (None表示综合提升)
```

```
        max_steps: 最多推荐几个知识点
```

```
    Returns:
```

```
        推荐学习的知识点列表 (按先后顺序)
```

```
    """
```

```
    MASTERY_THRESHOLD = 0.7 # 掌握度>70%视为已掌握
```

```
    # 找出未掌握的知识点 (mastery < threshold)
```

```
    unmastered = {
```

```
        kp for kp, mastery in mastery_levels.items()
```

```
        if mastery < MASTERY_THRESHOLD and kp in self.graph.nodes
```

```
    }
```

```
    if target_knowledge:
```

```
        # 目标导向: 找出到达目标知识点所需的前置知识
```

```
        unmastered &= self._get_prerequisites(target_knowledge)
```

```
        unmastered.add(target_knowledge)
```

```
    if not unmastered:
```

```
        return [] # 全部已掌握
```

```

# 拓扑排序（按依赖关系确定学习顺序）
topo_order = list(nx.topological_sort(self.graph))

# 过滤出未掌握的节点，保持拓扑顺序
path_ids = [n for n in topo_order if n in unmastered]

# 检查每个节点的前置条件是否满足
ready_to_learn = []
for kp_id in path_ids:
    prerequisites = list(self.graph.predecessors(kp_id))
    if all(mastery_levels.get(p, 0) >= MASTERY_THRESHOLD
           for p in prerequisites):
        ready_to_learn.append(kp_id)
        if len(ready_to_learn) >= max_steps:
            break

# 按学习难度和预计时间排序（优先推荐容易的知识点）
ready_to_learn.sort(
    key=lambda kp: (
        self.graph.nodes[kp].get('difficulty', 3),
        self.graph.nodes[kp].get('estimated_time_min', 30)
    )
)

return [
    KnowledgeNode(
        id=kp,
        name=self.graph.nodes[kp].get('name', kp),
        subject=self.graph.nodes[kp].get('subject', ''),
        difficulty=self.graph.nodes[kp].get('difficulty', 3),
        estimated_time_min=self.graph.nodes[kp].get('estimated_time_min', 20)
    )
    for kp in ready_to_learn
]

def _get_prerequisites(self, target: str) -> set:
    """获取目标知识点的所有前置知识（递归）"""
    return nx.ancestors(self.graph, target)

def get_progress_summary(
    self,
    mastery_levels: Dict[str, float],
    subject: Optional[str] = None
) -> Dict:
    """获取学习进度摘要"""
    nodes = [n for n in self.graph.nodes
              if subject is None or self.graph.nodes[n].get('subject') == subject]
    total = len(nodes)
    mastered = sum(1 for n in nodes if mastery_levels.get(n, 0) >= 0.7)
    return {
        'total': total,
        'mastered': mastered,
        'in_progress': sum(1 for n in nodes
                           if 0.3 <= mastery_levels.get(n, 0) < 0.7),
        'not_started': total - mastered - sum(
            1 for n in nodes if 0.3 <= mastery_levels.get(n, 0) < 0.7),
    }

```

```
        'completion_rate': mastered / total if total > 0 else 0.0,
    }
```

G.4 API接口补充

接口路径	方法	说明
/api/v1/analytics/class/{id}/realtime	GET	获取班级实时学情（WebSocket推送）
/api/v1/analytics/student/{id}/mastery	GET	获取学生知识点掌握度矩阵
/api/v1/analytics/student/{id}/path	GET	获取个性化学习路径推荐
/api/v1/analytics/student/{id}/mistakes	GET	获取错题分析报告
/api/v1/analytics/student/{id}/trend	GET	获取学习趋势折线图数据
/api/v1/analytics/class/{id}/heatmap	GET	获取知识点掌握度热力图
/api/v1/analytics/homework/{id}/stats	GET	获取作业统计分析
/api/v1/analytics/export/class/{id}	POST	导出班级PDF报告

附录G 补充技术规格

G.1 知识图谱构建与查询

G.1.1 学科知识点图谱结构

知识图谱采用有向无环图（DAG）建模学科知识点依赖关系：

```
// KnowledgeGraphService.java
@Service
public class KnowledgeGraphService {

    @Autowired
    private Neo4jTemplate neo4j; // 使用Neo4j图数据库

    /**
     * 查询某知识点的所有前置依赖（递归深度≤5）
     */
    public List<KnowledgeNode> findPrerequisites(String nodeId, int maxDepth) {
        String cypher =
            "MATCH path = (target:KnowledgeNode {id: $nodeId})" +
            "<-[:REQUIRES*1.."+ maxDepth + "]- (prereq:KnowledgeNode) " +
            "RETURN DISTINCT prereq " +
            "ORDER BY length(path) ASC";
```

```

        return neo4j.query(cypher,
            Map.of("nodeId", nodeId), KnowledgeNode.class);
    }

    /**
     * 查找两个知识点之间的最短学习路径
     */
    public List<KnowledgeNode> shortestLearningPath(String fromId, String toId) {
        String cypher =
            "MATCH path = shortestPath(" +
            " (from:KnowledgeNode {id: $fromId})-[:REQUIRES*]->(to:KnowledgeNode {id: " +
            "$toId}))" +
            ") RETURN nodes(path) AS nodes";

        return neo4j.queryForObject(cypher,
            Map.of("fromId", fromId, "toId", toId),
            result -> (List<KnowledgeNode>) result.get("nodes"));
    }

    /**
     * 获取学生当前可学习的知识点（前置条件已掌握）
     */
    public List<KnowledgeNode> getReadyToLearn(String studentId, String subjectId) {
        String cypher =
            "MATCH (s:Student {id: $studentId})-[:MASTERED]->(mastered:KnowledgeNode) " +
            "MATCH (candidate:KnowledgeNode {subject: $subjectId}) " +
            "WHERE NOT (s)-[:MASTERED]->(candidate) " + // 尚未掌握
            "AND NOT (candidate)-[:REQUIRES]->(:KnowledgeNode " +
            " WHERE NOT (s)-[:MASTERED]->()) " + // 所有前置已掌握
            "RETURN candidate ORDER BY candidate.difficulty ASC LIMIT 10";

        return neo4j.query(cypher,
            Map.of("studentId", studentId, "subjectId", subjectId),
            KnowledgeNode.class);
    }
}

```

G.2 学情报告PDF生成

G.2.1 JasperReports模板引擎

```

// ReportGenerationService.java
@Service
public class ReportGenerationService {

    private static final String TEMPLATE_DIR = "/templates/reports/";

    @Autowired
    private StudentAnalyticsService analyticsService;

    public byte[] generateStudentReport(String studentId,

```

```

                                LocalDate startDate,
                                LocalDate endDate) {

// 1. 收集报告数据
StudentReportData data = buildReportData(studentId, startDate, endDate);

// 2. 加载JasperReport模板
InputStream templateStream = getClass().getResourceAsStream(
    TEMPLATE_DIR + "student_report.jrxml");
JasperReport jasperReport = JasperCompileManager.compileReport(templateStream);

// 3. 填充数据
Map<String, Object> params = new HashMap<>();
params.put("studentName", data.getStudentName());
params.put("reportPeriod", data.getReportPeriod());
params.put("masteryRate", data.getMasteryRate());
params.put("CHART_DATA", new JRBeanCollectionDataSource(data.getChartItems()));
params.put("MISTAKE_DATA", new JRBeanCollectionDataSource(data.getMistakes()));

JasperPrint jasperPrint = JasperFillManager.fillReport(
    jasperReport, params, new JREmptyDataSource());

// 4. 导出为PDF
return JasperExportManager.exportReportToPdf(jasperPrint);
}

private StudentReportData buildReportData(String studentId,
                                           LocalDate start,
                                           LocalDate end) {

    StudentReportData data = new StudentReportData();
    data.setStudentName(studentRepo.findById(studentId).getName());
    data.setReportPeriod(start + " 至 " + end);

    // 各学科掌握度
    data.setMasteryRate(analyticsService.getMasteryRateBySubject(
        studentId, start, end));

    // 学习时长趋势（按周聚合）
    data.setChartItems(analyticsService.getLearningTimeTrend(
        studentId, start, end, "WEEKLY"));

    // 高频错题TOP10
    data.setMistakes(analyticsService.getTopMistakes(studentId, 10));

    return data;
}
}

```

G.3 实时学情流式处理

G.3.1 Flink CEP复杂事件检测

```

// LearningEventCEP.java - 复杂事件处理规则
public class LearningEventCEP {

```

```

/**
 * 检测"连续3次答错"事件:
 * 触发后向教师推送预警
 */
public static PatternStream<AnswerEvent> consecutiveWrongAnswers(
    DataStream<AnswerEvent> stream) {

    Pattern<AnswerEvent, ?> pattern = Pattern
        .<AnswerEvent>begin("first_wrong")
        .where(e -> !e.isCorrect())
        .next("second_wrong")
        .where(e -> !e.isCorrect())
        .next("third_wrong")
        .where(e -> !e.isCorrect())
        .within(Time.minutes(10)); // 10分钟内

    return CEP.pattern(stream.keyBy(AnswerEvent::getStudentId), pattern);
}

/**
 * 检测"长时间无操作"事件:
 * 超过5分钟未提交任何答案
 */
public static DataStream<IdleAlert> detectIdleStudents(
    DataStream<AnswerEvent> stream) {

    return stream
        .keyBy(AnswerEvent::getStudentId)
        .window(TumblingEventTimeWindows.of(Time.minutes(5)))
        .aggregate(new CountAggregator())
        .filter(count -> count == 0)
        .map(count -> new IdleAlert(count.getStudentId()));
}
}

```

G.4 数据导出与报表功能

G.4.1 Excel多Sheet导出

```

// ExcelExportService.java
@Service
public class ExcelExportService {

    public byte[] exportClassAnalytics(String classId,
                                       LocalDate startDate,
                                       LocalDate endDate) throws IOException {
        try (XSSFWorkbook workbook = new XSSFWorkbook()) {
            // Sheet1: 班级总览
            createClassOverviewSheet(workbook, classId, startDate, endDate);
            // Sheet2: 知识点掌握度矩阵
            createMasteryMatrixSheet(workbook, classId);
            // Sheet3: 错题统计

```

```

        createMistakeAnalysisSheet(workbook, classId, startDate, endDate);
        // Sheet4: 学生排名
        createStudentRankingSheet(workbook, classId, startDate, endDate);

        ByteArrayOutputStream out = new ByteArrayOutputStream();
        workbook.write(out);
        return out.toByteArray();
    }
}

private void createMasteryMatrixSheet(XSSFWorkbook wb, String classId) {
    Sheet sheet = wb.createSheet("知识点掌握度");

    // 热力图样式: 绿色=掌握, 黄色=部分掌握, 红色=未掌握
    XSSFCellStyle greenStyle = createColorStyle(wb, new XSSFColor(
        new byte[]{(byte)144, (byte)238, (byte)144}, null));
    XSSFCellStyle yellowStyle = createColorStyle(wb, new XSSFColor(
        new byte[]{(byte)255, (byte)255, (byte)0}, null));
    XSSFCellStyle redStyle = createColorStyle(wb, new XSSFColor(
        new byte[]{(byte)255, (byte)99, (byte)71}, null));

    List<Student> students = studentRepo.findById(classId);
    List<KnowledgeNode> nodes = knowledgeRepo.findByGrade(
        classRepo.findById(classId).getGrade());

    // 表头: 知识点名称
    Row header = sheet.createRow(0);
    header.createCell(0).setCellValue("学生姓名");
    for (int j = 0; j < nodes.size(); j++) {
        header.createCell(j + 1).setCellValue(nodes.get(j).getName());
    }

    // 数据行
    for (int i = 0; i < students.size(); i++) {
        Row row = sheet.createRow(i + 1);
        row.createCell(0).setCellValue(students.get(i).getName());

        for (int j = 0; j < nodes.size(); j++) {
            double mastery = masteryService.getMastery(
                students.get(i).getId(), nodes.get(j).getId());

            Cell cell = row.createCell(j + 1);
            cell.setCellValue(String.format("%.0f%%", mastery * 100));

            if (mastery >= 0.8) cell.setCellStyle(greenStyle);
            else if (mastery >= 0.5) cell.setCellStyle(yellowStyle);
            else cell.setCellStyle(redStyle);
        }
    }

    // 自动列宽
    for (int i = 0; i <= nodes.size(); i++) sheet.autoSizeColumn(i);
}
}

```

附录H 补充技术规格

H.1 错题智能分析

H.1.1 错误模式聚类

```
// MistakePatternAnalyzer.java
@Service
public class MistakePatternAnalyzer {

    /**
     * 对学生的错题进行K-Means聚类，发现共性错误模式
     */
    public List<MistakeCluster> clusterMistakes(String studentId) {
        List<MistakeRecord> mistakes = mistakeRepo.findByStudentId(studentId);
        if (mistakes.size() < 3) return List.of(); // 样本不足

        // 提取特征向量：[知识点ID， 错误类型， 难度等级]
        List<double[]> features = mistakes.stream()
            .map(m -> new double[]{
                knowledgeGraph.getNodeIndex(m.getKnowledgeNodeId()),
                m.getErrorType().ordinal(),
                m.getDifficultyLevel()
            })
            .collect(Collectors.toList());

        // K-Means聚类 (K=3)
        KMeansPlusPlusClusterer<DoublePoint> clusterer =
            new KMeansPlusPlusClusterer<>(3, 100);
        List<CentroidCluster<DoublePoint>> clusters =
            clusterer.cluster(features.stream()
                .map(DoublePoint::new)
                .collect(Collectors.toList()));

        return clusters.stream().map(cluster -> {
            MistakeCluster mc = new MistakeCluster();
            mc.setCentroid(cluster.getCenter().getPoint());
            mc.setSize(cluster.getPoints().size());
            mc.setDominantPattern(analyzeDominantPattern(cluster));
            mc.setRecommendedContent(recommendContent(mc.getDominantPattern()));
            return mc;
        }).collect(Collectors.toList());
    }

    private String analyzeDominantPattern(CentroidCluster<DoublePoint> cluster) {
        // 分析聚类中最常见的错误模式
        double[] centroid = cluster.getCenter().getPoint();
        int errorTypeIdx = (int) Math.round(centroid[1]);
        return ErrorType.values()[errorTypeIdx].getDescription();
    }
}
```

H.2 自适应练习推送

```
// AdaptivePracticeService.java
@Service
public class AdaptivePracticeService {

    /**
     * 基于当前学生掌握度，自动选择下一道练习题
     * 目标：选择掌握度在40-70%之间的知识点对应题目（最佳学习区间）
     */
    public Question selectNextQuestion(String studentId, String subjectId) {
        // 获取所有知识点掌握度
        Map<String, Double> masteryMap = bktService.getAllMastery(studentId, subjectId);

        // 筛选在适合练习区间内的知识点（40%≤掌握度≤70%）
        List<String> candidateNodes = masteryMap.entrySet().stream()
            .filter(e -> e.getValue() >= 0.4 && e.getValue() <= 0.7)
            .sorted(Map.Entry.comparingByValue()) // 优先掌握度较低的
            .map(Map.Entry::getKey)
            .limit(5)
            .collect(Collectors.toList());

        if (candidateNodes.isEmpty()) {
            // 无合适区间，选择掌握度最低的知识
            candidateNodes = masteryMap.entrySet().stream()
                .min(Map.Entry.comparingByValue())
                .map(e -> List.of(e.getKey()))
                .orElse(List.of());
        }

        if (candidateNodes.isEmpty()) {
            return questionRepo.findRandom(subjectId);
        }

        // 从候选知识点中随机选题
        String targetNode = candidateNodes.get(
            (int)(Math.random() * candidateNodes.size()));

        // 排除最近已做过的题目（避免重复）
        Set<String> recentQuestions = practiceHistoryRepo
            .findRecentQuestionIds(studentId, 20);

        return questionRepo.findByKnowledgeNode(targetNode, recentQuestions);
    }
}
```

H.3 学习报告定时生成

```
// ScheduledReportGenerator.java
@Component
public class ScheduledReportGenerator {

    @Autowired
```

```

private ReportGenerationService reportService;

@Autowired
private NotificationService notificationService;

// 每周一早上8点生成上周学情报告
@scheduled(cron = "0 0 8 * * MON")
public void generateWeeklyReports() {
    log.info("开始生成每周学情报告...");
    LocalDate endDate = LocalDate.now().minusDays(1);
    LocalDate startDate = endDate.minusWeeks(1);

    List<String> activeStudentIds = studentRepo.findActiveStudentIds();

    activeStudentIds.parallelStream().forEach(studentId -> {
        try {
            byte[] pdf = reportService.generateStudentReport(
                studentId, startDate, endDate);

            // 上传到OSS
            String reportUrl = ossService.upload(
                String.format("reports/%s/%s_weekly.pdf", studentId, endDate),
                pdf);

            // 推送通知给学生和家长
            notificationService.sendReportReady(studentId, reportUrl, "weekly");

        } catch (Exception e) {
            log.error("生成报告失败: student={}", studentId, e);
        }
    });

    log.info("每周报告生成完成, 共{}份", activeStudentIds.size());
}

// 每天凌晨2点更新知识点掌握度
@scheduled(cron = "0 0 2 * * *")
public void updateMasteryScores() {
    log.info("开始批量更新掌握度评分...");
    bktService.batchUpdateAllStudents();
}
}

```

附录I 补充技术规格

I.1 学生行为数据采集

```

// BehaviorTrackingService.java
@Service
public class BehaviorTrackingService {

```

```

@Autowired
private KafkaTemplate<String, BehaviorEvent> kafkaTemplate;

/**
 * 记录学生行为事件（异步发送到Kafka，不影响主流程）
 */
@Async
public void track(String studentId, BehaviorEventType type, Map<String, Object>
data) {
    BehaviorEvent event = BehaviorEvent.builder()
        .studentId(studentId)
        .type(type)
        .data(data)
        .timestamp(Instant.now())
        .sessionId(getCurrentSessionId())
        .deviceInfo(getDeviceInfo())
        .build();

    kafkaTemplate.send("student-behavior-events", studentId, event);
}

// 常用埋点方法
public void trackHomeworkStart(String studentId, String homeworkId) {
    track(studentId, BehaviorEventType.HOMEWORK_START, Map.of(
        "homework_id", homeworkId,
        "start_time", Instant.now().toString()
    ));
}

public void trackQuestionAnswer(String studentId, String questionId,
                                boolean correct, long timeTakenMs) {
    track(studentId, BehaviorEventType.QUESTION_ANSWER, Map.of(
        "question_id", questionId,
        "correct", correct,
        "time_taken_ms", timeTakenMs
    ));
}

public void trackStudySession(String studentId, long durationMs,
                               String subjectId) {
    track(studentId, BehaviorEventType.STUDY_SESSION_END, Map.of(
        "duration_ms", durationMs,
        "subject_id", subjectId,
        "focus_score", calculateFocusScore(studentId, durationMs)
    ));
}

private double calculateFocusScore(String studentId, long durationMs) {
    // 基于答题速度和正确率计算专注度评分
    return Math.min(1.0, durationMs / 1800000.0); // 30分钟满分
}
}

```

1.2 数据隐私保护

```
// PrivacyDataMasker.java
@Component
public class PrivacyDataMasker {

    /**
     * 对敏感字段进行脱敏处理
     * 用于数据导出和API响应中保护用户隐私
     */
    public StudentDTO maskStudentData(Student student, boolean isParent) {
        StudentDTO dto = StudentDTO.fromEntity(student);

        if (!isParent) {
            // 非家长查看时隐藏部分信息
            dto.setPhone(maskPhone(student.getPhone()));
            dto.setParentName(maskName(student.getParentName()));
        }

        // 始终隐藏身份证号后半段
        if (student.getIdCard() != null) {
            dto.setIdCard(student.getIdCard().substring(0, 6) + "*****");
        }

        return dto;
    }

    public String maskPhone(String phone) {
        if (phone == null || phone.length() < 7) return "***";
        return phone.substring(0, 3) + "****" + phone.substring(7);
    }

    public String maskName(String name) {
        if (name == null || name.isEmpty()) return "***";
        if (name.length() == 2) return name.charAt(0) + "*";
        return name.charAt(0) + "*".repeat(name.length() - 2) +
            name.charAt(name.length() - 1);
    }
}
```

1.3 版本历史

版本号	发布日期	变更说明	负责人
V1.0.0	2024-01-15	初始版本，实现基础学情数据采集与展示	研发团队
V1.1.0	2024-03-01	引入BKT贝叶斯知识追踪算法	算法组
V1.2.0	2024-04-20	集成Flink实时流处理，学情数据实时更新	大数据组
V1.3.0	2024-06-15	新增知识图谱，支持学习路径DAG推荐	AI组
V1.4.0	2024-08-01	接入ClickHouse，历史数据分析查询提速10倍	数据组

版本号	发布日期	变更说明	负责人
V1.5.0	2024-10-15	新增JasperReports PDF报告生成功能	研发团队
V1.6.0	2024-12-01	错题K-Means聚类分析，个性化练习推送	AI组

I.4 术语表

术语	英文缩写	说明
贝叶斯知识追踪	BKT	基于隐马尔可夫模型评估学生知识掌握度
知识图谱	KG	有向无环图描述学科知识点依赖关系
实时流处理	Stream Processing	Flink处理毫秒级学情事件数据流
列式存储	ClickHouse	面向分析场景的高性能OLAP数据库
间隔重复	Spaced Repetition	Leitner算法优化复习时间间隔
学习路径	Learning Path	DAG算法推荐个性化知识点学习顺序

本文档版权归深圳自然写科技有限公司所有，所有技术细节与源代码对应关系仅用于软件著作权登记鉴别。