服务器日志异常检测系统设计

1. 项目背景

原始日志很长且定位目标字段耗时，日志中混杂多种内容干扰排查。主要体现如下：

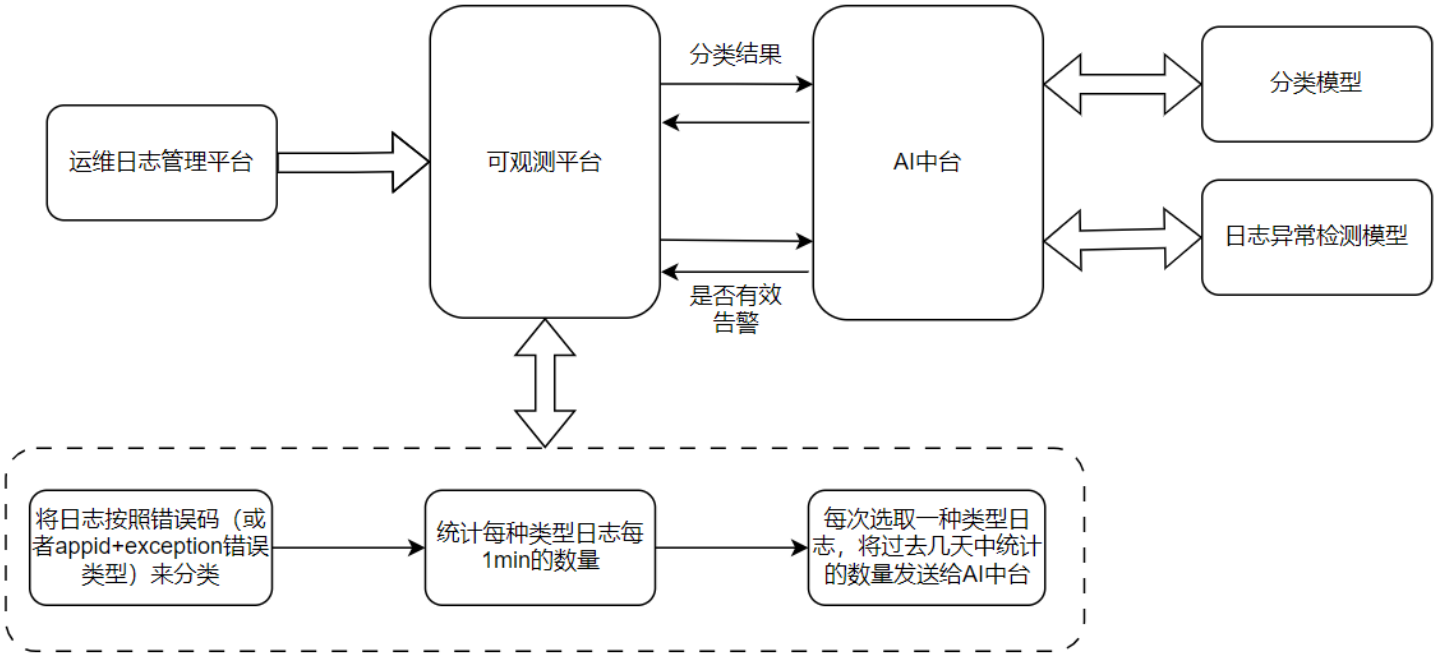
1. **部分错误码没有配置告警：**错误码完全依赖人工，开发必须告知运维才会配置告警，而有些错误码开发没有告知运维的。
2. **部分错误码不可靠：**有些错误码是每天发现，且无风险隐患的，影响运维排查日志。
3. **exception日志难以快速定位问题：**ERROR级别的日志可能会出现没有错误码，只有exception的情况，这类日志都会归为同一类型错误，即exception日志混合了无风险隐患的和有风险隐患的日志。
4. **exception日志很长，定位问题耗时：**日志的exception一般是java代码的堆栈信息，需要解析堆栈信息才能理解是什么类型的错误。


```
{"timeStamp":"2018-03-2116:56:40.363",”authorizeAcc”:”xxxx”,”traceId”:””,”invocationId”:””,”
spanId”:””,”
parentSpanId”:””,"requestId":"65209104891405","pid":"13395","level":"ERROR","threadName":
"KAF_ASYNC-
1","loggerName":"com.youcash.monitor.service.kafkaListener.complete.applyLoanContainer","ap
pID":"AA0","type":"M","Messagecode":"ESAA05XX000","Message":"系统异
常","responseTime":"788",”SeqNo”:””,”BASY-ID”:””,”SEQ-ID”:””,"content":
{"phoneNo":"15072307231","OrderNo":"BC20180329000068","Acctid":"6217995804694946494"
,"IdNo":"421182199303216213",”CardNo”:””,”Name”:””,”CardType”:””,”
ProductNo”:””},"exception":"\norg.springframework.dao.DataIntegrityViolationException:
\n### Error updating database.  Cause: java.sql.SQLIntegrityConstraintViolationException:
ORA-02291: 违反完整约束条件 (PSBCAP.FK_REJECTRECORD_APPNUM) - 未找到父项关键字
\n\n### The error may involve com.youcash.las.apply.dao.LoanRejectRecordMapper.insert-
Inline\n### The error occurred while setting parameter\n\tat "}
```


具体现状：

1. 运维每天告警的日志达到千万条，如20250213共有10778457条error日志，运维人员无法逐条分析，需要模型定位出有效的error日志。
2. 无错误码，有exception日志都归为一类，且每天量级到达200w条，如20250213共有2405530条，出现少数严重级的告警日志时，无法快速定位。
3. 有错误码的日志进行分级监控，如部分错误码日志出现则排查，部分错误码日志在5分钟内出现5次则排查等。
4. 无错误码，有exception日志则直接跳过。

2. 项目工作流



1. 可观测平台从运维日志管理平台获取日志数据
2. 可观测平台将日志按照错误码（或者appid+exception错误类型）来分类，并统计每种错误码的每1min的数量，并保存下来。如果出现新的类型日志，则直接抛出告警并攒数。
3. 每次选取一种类型日志，将过去几天（3或7天等）中的统计数量发送给AI中台，AI中台调用分类模型，将分类结果返回给可观测平台并保存（每种类型日志的调用频率为1天以上，甚至是1周或1月）。
4. 每次选取一种类型日志，将过去几天（3，7或14天等）中的统计数量发送给AI中台，然后再调用对应的日志异常检测模型，返回是否有效警告（实时，每5分钟一次）。

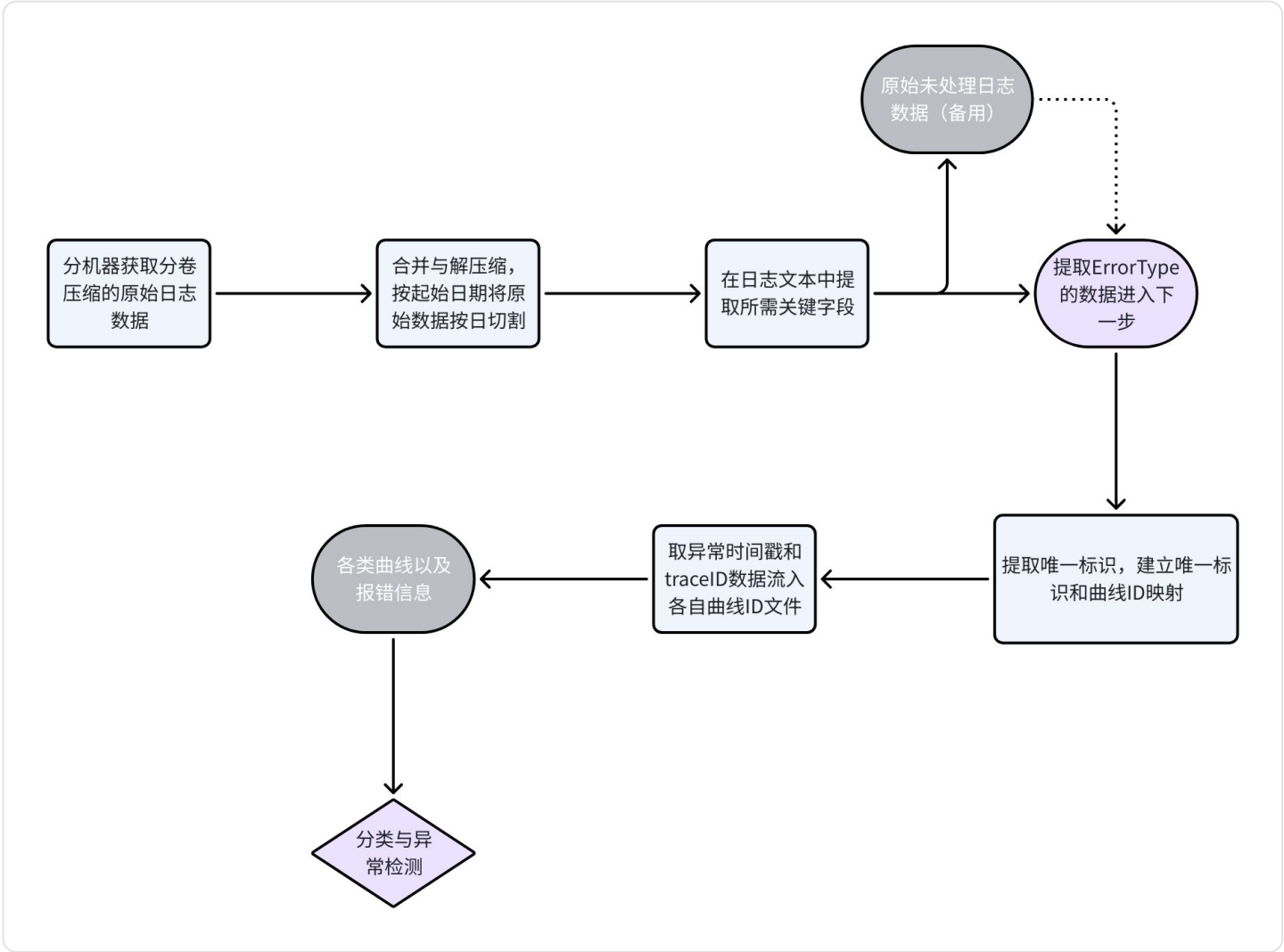
3. 流程设计

3.1 数据预处理

原始日志信息log_msg很复杂，如下所示，从中提取出**timeStamp、appID、traceId、LoggerName、Messagecode、exception**字段进行下一步，使用正则表达式提取**exception**中错误类型信息**ErrorType**，采用“**appID|||Messagecode|||LoggerName|||ErrorType**”作为日志记录的唯一标识符，将所有日志记录按唯一标识符划分汇总到每一种日志中形成日志异常曲线。

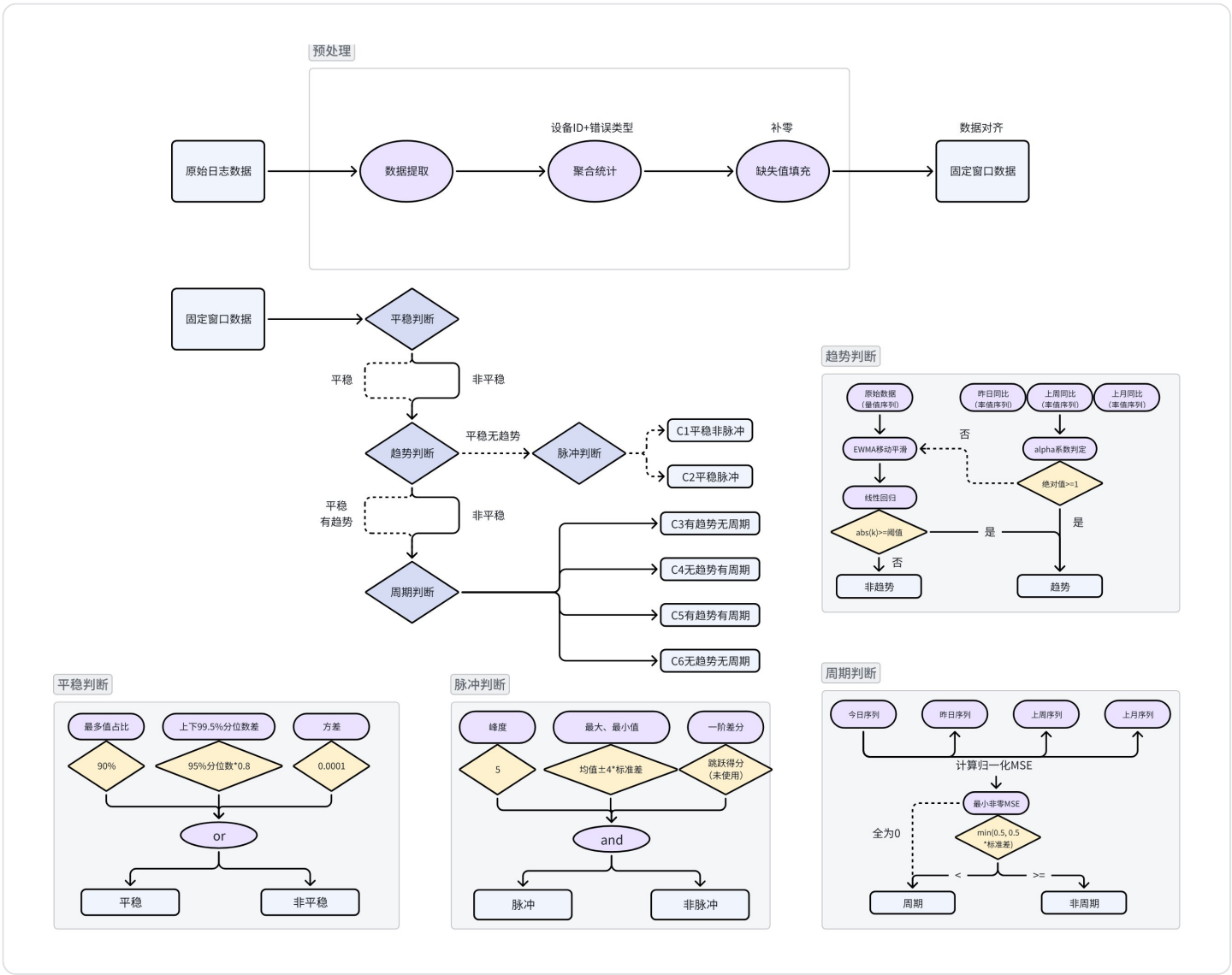
其中，对Exception匹配正则表达式：**r'\n(.*?(?:Error|Exception).*?):'** 得到ErrorType，如上方示例日志提取出的唯一curve_ID为：

> 'AA0|||ESAA05XX000|||com.youcash.monitor.service.kafkaListener.complete.applyLoanContainer|||org.springframework.dao.DataIntegrityViolationException'

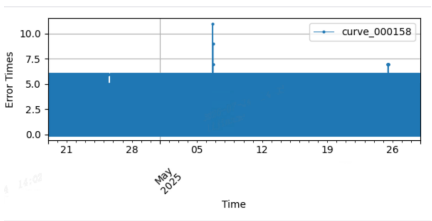
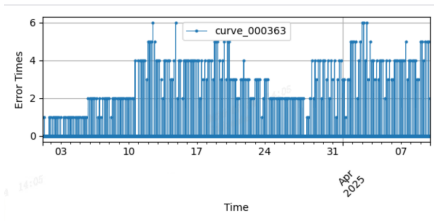
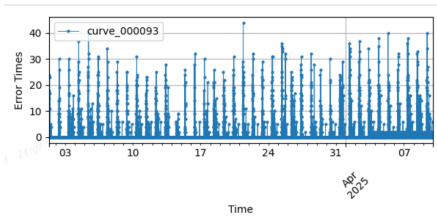
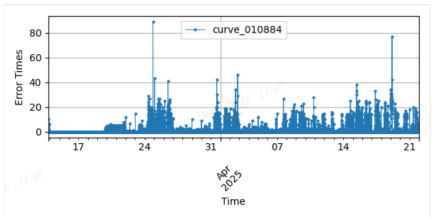


3.2 曲线分类

依据三种特征将时间序列分为不同的类别：平稳性，趋势性，周期性。

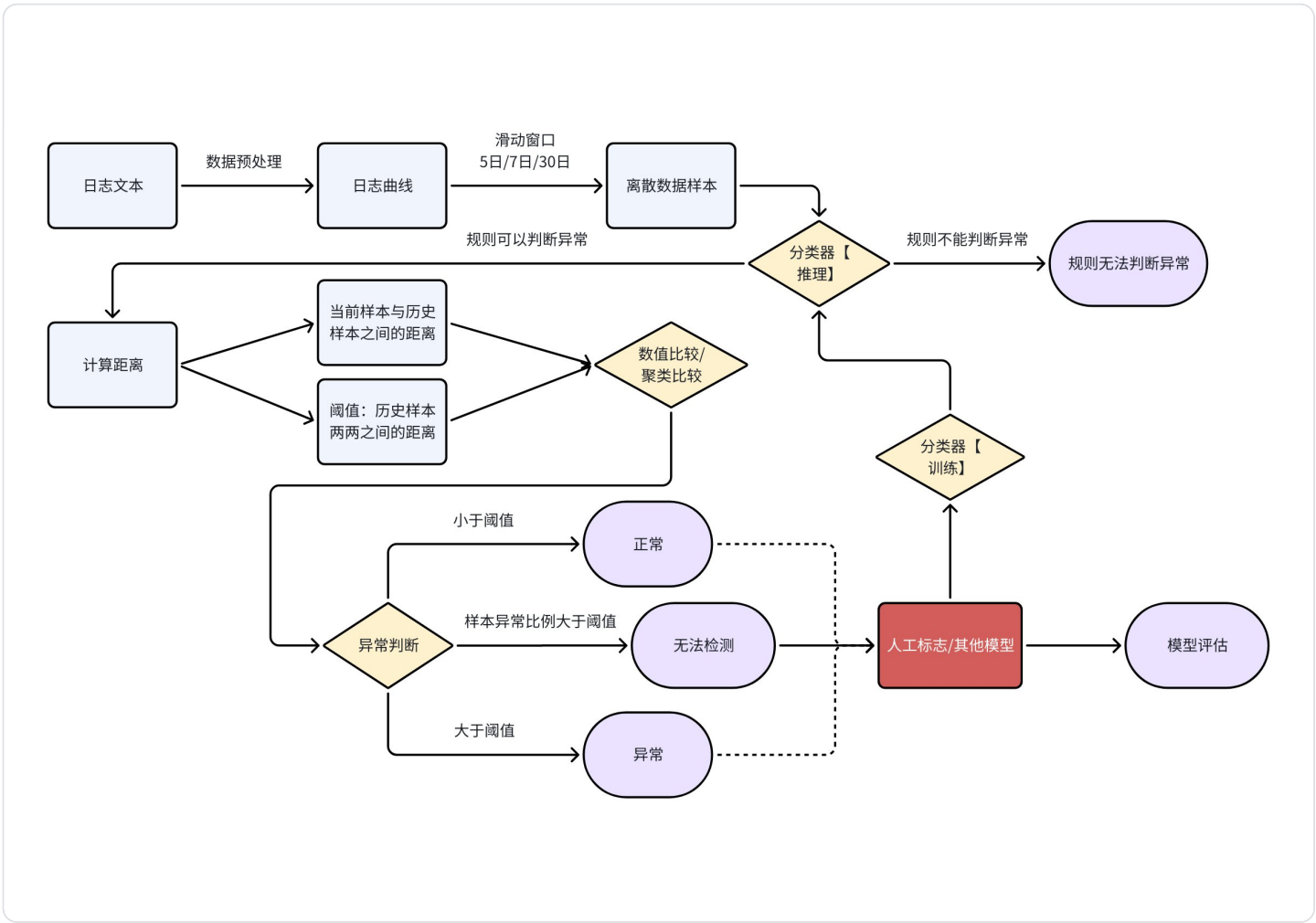


预计提取出曲线标准形式示例如下：

C1纯平稳型	C2脉冲型	C4周期型
		
C6随机型	**C3趋势型**	**C5周期趋势型**
		

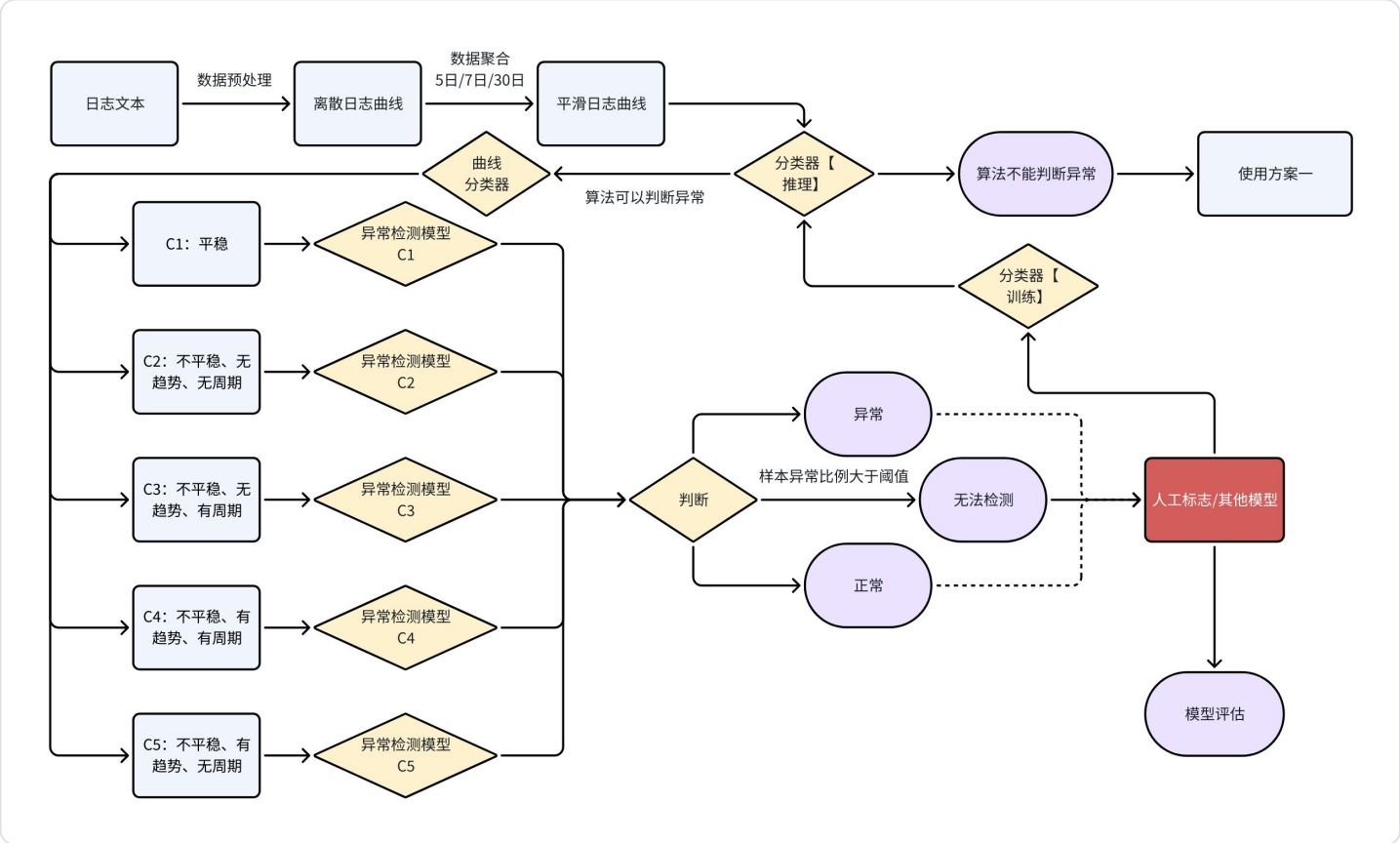
试运行发现，数据中趋势型以及周期趋势型极少，只有个别曲线片段存在弱趋势特征，因此将C3趋势型和C5周期趋势型并入C6随机型中进行统一处理。

3.3 简单规则异常检测



异常判断规则的核心为：计算当前一段数据（如连续5个数据点）与历史数据（如历史连续5个数据点）的相似度，相似度过低意味着当前曲线的波动模式没有在历史数据中出现过，即判定为异常。衡量相似度依靠计算两段曲线之间的距离，考虑尝试多种距离方案选取性能和速度综合最优。

3.4 统计模型异常检测



3.5 统计异常检测器设计

3.5.1 C1平稳型异常检测器

略

3.5.2 C2脉冲型异常检测器

略

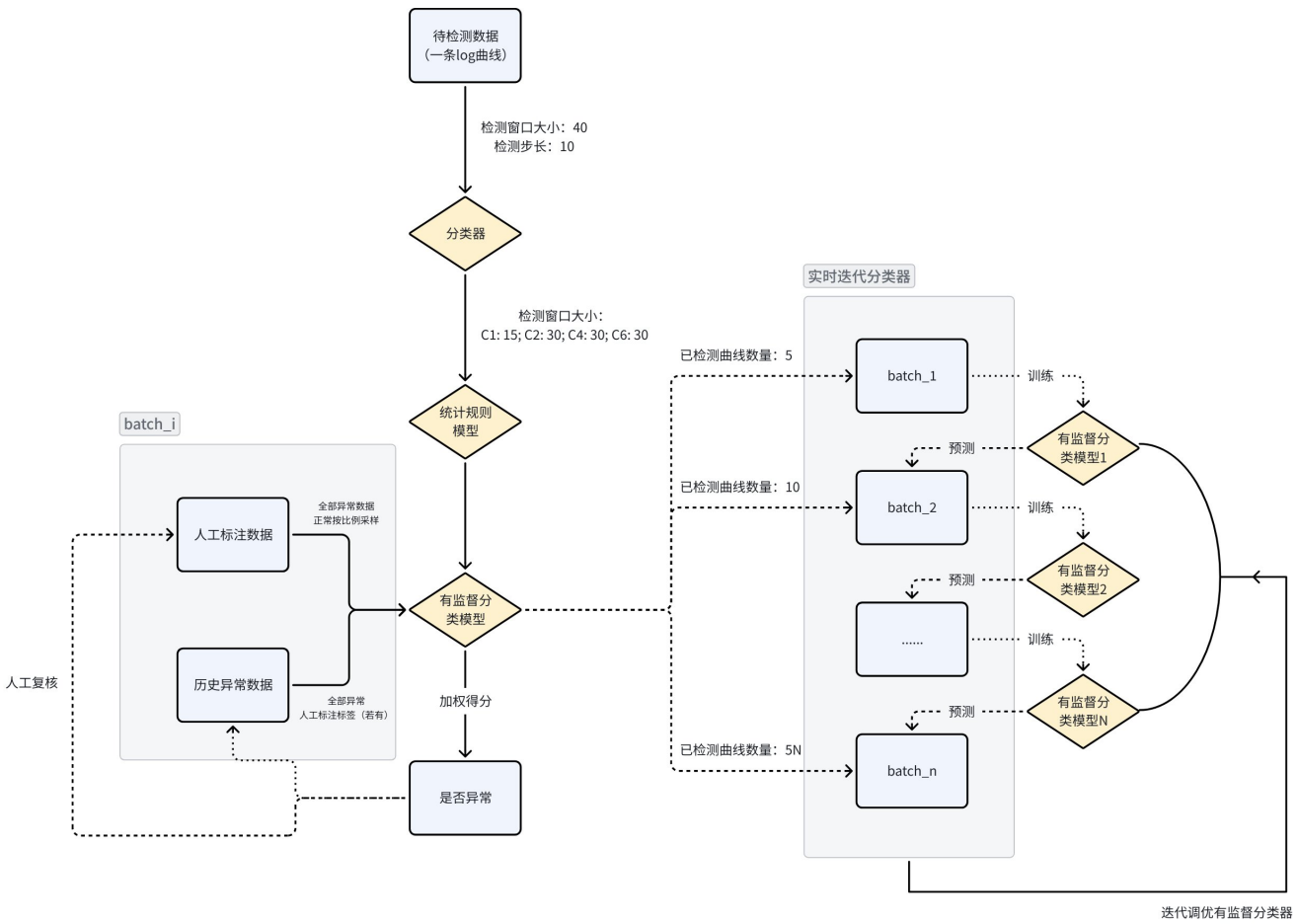
3.5.3 C4周期型异常检测器

略

3.5.4 C6随机型异常检测器

略

3.6 批量预打标流程设计



通过搜集当前统计模型判断为异常的点和历史已标注数据训练有监督二分类模型，用来学习一个通用的异常模式并对后续统计模型的判断结果进行修正，实现自迭代训练流程，并且引入了历史高保真标注数据对结果进行修正，在这种情况下可以调整统计模型放松对精确率的要求，优先保证高召回率尽可能将异常点筛选出来，计算特征后交由精确率更高的有监督二分类模型进行修正提高精确率。

4. 主要结果

4.1 曲线统计结果（部分）

6月份日志数量统计结果如下：

MsgCode/Log/Err	曲线数量	日志数量	MessageCode种类	LoggerName种类	exception种类
Y/Y/Y	1236	79320520	695	403	97
N/Y/Y	2822	23766837	--	1236	274
Y/N/Y	0	0	0	--	0
Y/Y/N	438	23913844	336	255	--

N/N/Y	0	0	--	--	0
N/Y/N	1428	188605940	--	821	--
Y/N/N	1	699	1	--	--
N/N/N	3	133546	--	--	--
总计	5928	**315741386**	986	2381	295

4.2 曲线分类结果（部分）

混淆矩阵：

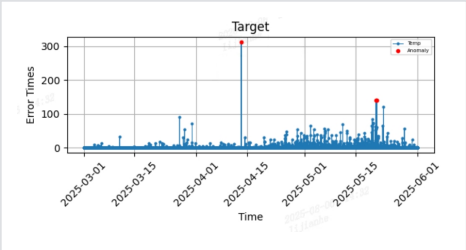
true|pre	C1	C2	C4	C6
C1	10766	242	45	27
C2	141	14533	8	94
C4	16	9	552	28
C6	17	58	41	271

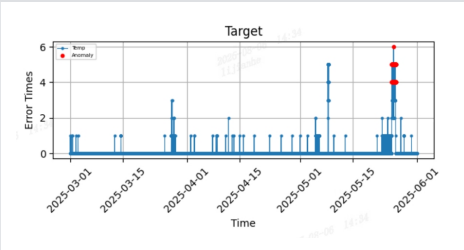
各类指标：

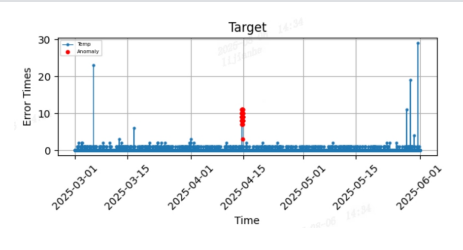
	precision	recall	f1-score	support
C1	94.41%	97.17%	97.78%	11080
C2	97.92%	98.36%	98.14%	14776
C4	85.45%	91.24%	88.25%	605
C6	64.52%	70.03%	67.16%	387
Accuracy	--	--	**97.30%**	26848
Macro avg	85.58%	89.20%	87.83%	26848
Weighted avg	97.36%	97.30%	97.32%	26848

采样40日窗口大小、10天步长进行分类，分类准确率达**97.30%**，平均f1达到**87.83%**

4.3 异常检测结果（部分）







	precision	recall	f1-score	support
异常	97.20%	96.72%	96.96%	13104696
Accuracy	--	--	99.98%	53038

5. 后续改进方向

1. 由于项目时间紧急，以及本人能力不足，目前仅实现对纯平稳型曲线和脉冲型曲线的简单异常检测，项目上线后作为第一版Base模型，便于后续优化对比；
2. 等待项目迭代运行一段时间后搜集足够的异常数据，后续用来进行有监督的机器学习训练；
3. 本项目保留了每条日志的traceID，后续可以通过traceID关联不同服务节点的日志，形成完整的异常日志链条，方便运维定位异常源头。

6. 参考文献

1. https://tech.meituan.com/2023/12/22/aiops-based-incident-management.html
2. https://tech.meituan.com/2017/04/21/order-holtwinter.html
3. https://tech.meituan.com/2020/10/15/mt-aiops-horae.html
4. https://cloud.tencent.com/developer/article/1670322
5. https://github.com/Tencent/Metis/
6. https://tech.meituan.com/2018/11/01/cat-in-depth-java-application-monitoring.html
7. https://www.alibabacloud.com/blog/alibaba-engineers-have-worked-out-loads-of-methods-to-detect-anomalies_595452
8. https://cloud.tencent.com/developer/article/1538908?from_column=20421&from=20421
9. https://juejin.cn/post/6844903743280906254
10. https://tech.meituan.com/2023/04/20/traceid-google-dapper-mtrace.html
11. https://tech.meituan.com/2022/09/01/database-monitoring-based-on-ai.html