arXiv:2303.06094v1 [cs.DB] 10 Mar 2023

在数据损坏时快速行动

为机器学习流水线实施自动数据验证系统

Shreya Shankar* Labib Fawaz 加州大学伯克利分校 Meta shreyashankar@berkeley.edu labibfawaz@meta.com

Karl Gyllstrom Meta gylls@meta.com

Aditya G. Parameswaran 加州大学伯克利分校 adityagp@berkeley.edu

摘要

生产流水线中的机器学习（ML）模型经常会基于大型且持续增长的数据集的最新分区重新训练。由于工程缺陷，此类数据集中的分区几乎总是包含一些损坏特征；因此，在下游机器学习模型准确率下降之前，检测数据问题并阻止重新训练至关重要。然而，很难判断一个分区何时损坏到足以阻止重新训练的程度。阻止得过于频繁，会导致生产环境中的模型快照过时；阻止得太少，则会导致生产环境中的模型快照失效。

本文介绍了 Meta 为机器学习流水线实现的自动数据验证系统。我们采用一种称为分区汇总（PS）的数据验证方法：使用数据质量指标汇总每个基于时间戳的数据分区，并对比这些摘要以检测损坏的分区。我们介绍如何将 PS 应用于多种数据验证方法，并比较它们各自的优缺点。由于这些方法单独使用时都无法满足我们对检测损坏的高精确率和高召回率的需求，我们设计了 gate，这是一种兼具高精确率和高召回率的数据验证方法。在一项使用 Instagram 数据的案例研究中，gate 的精确率平均提升了 2.1×，优于基线。最后，我们讨论了为 Meta 的生产机器学习流水线实现数据验证系统所得的经验教训。

1 引言

输入数据中的错误会对机器学习（ML）的性能产生负面影响，因此需要对机器学习流水线中的数据进行验证 [5, 8, 33] [8] 。

Breck 等人指出：“这个问题的重要性怎么强调都不为过，尤其是在生产流水线中”，因为机器学习模型经常会基于最新的数据分区重新训练——这些分区既包括底层输入特征，也包括相应的预测 [62]。例如，假设 Facebook 应用的最新版本无法播放音频，那么新闻动态排序模型的性能可能会受到影响，因为已更新应用的用户所对应的音频相关特征会遭到破坏。输入数据的某个分区（即某一天）遭到破坏，显而易见的后果是该分区上的模型性能或准确率偏低；而不那么显而易见、影响也更持久的后果是，在该分区上重新训练的模型也会受到破坏，从而引发难以恢复的错误循环。在我们的例子中，修复音频问题后，机器学习性能实际上又会出现偏差，因为模型已经基于遭到破坏的音频特征（例如表示声音是否静音的二元指示变量）重新训练。因此，本文关注的问题是 **自动验证数据，以检测生产机器学习流水线中的问题，在模型重新训练之前**。

在 Meta 这样的规模下，要在生产机器学习流水线中的数据错误导致下游准确率下降之前将其检测出来非常困难。数据分区可能非常大（即数个拍字节），并包含数万个特征（即模型的输入数据）。特征数量如此之多，工程师几乎无法控制模型开发者之外的工程缺陷，因此总会有一些特征遭到损坏——尤其是在许多组织（如 Meta）中，特征生成管道与机器学习模型的调用相互分离时 [54, 62]。此外，尚不清楚数据损坏达到何种阈值才会对准确率产生明显影响 [43]。机器学习模型实际上能够容忍一些错误：例如，如果某个不重要的特征出错，变成空值或垃圾值，机器学习性能（即预测准确率）通常不会改变。很难知道一个分区损坏到何种程度才会显著影响性能。此外，不同数据处理管道的错误容忍度各不相同，因此每条数据处理管道都需要值班工程师调查生产环境中的模型快照是否遭到损坏，以及是否应回滚到较早的模型版本 [54]。数据验证系统应在必要时向 Meta 的值班工程师发出警报，以 gate 或阻止将模型快照部署到生产环境，而不是等到后续预测的质量下降后才采取行动。

以往的数据验证技术无法解决我们的问题。 关于机器学习数据验证的研究，对于如何应对上述挑战所提供的见解有限，相关方法要么*(i)*过度依赖人工输入，要么*(ii)* 粒度过于粗糙，要么*(iii)* 产生过多误报。以*(i)*为例，Schelter 等人 [51]提出了多种通用的大规模数据验证技术；然而，这些技术在实际操作中难以扩展，因为它们要求我们的工程师为每个特征枚举并微调约束。以*(ii)*为例，Breck 等人 [8]为 Google 的生产机器学习流水线提出了模式验证 技术。这些技术会根据模式检查每个元组，模式由类型检查和定义较宽松的边界（例如，年龄相关特征的非负值）组成。模式验证不可或缺，但并不足够；大多数机器学习流水线都进行了某种模式验证，但仍然存在许多数据损坏问题。Breck 等人 [8] 还提出了一种量化特征分布差异的技术，例如使用推土机距离。然而，关于*(iii)*，Breck 等人 [8] 和我们的研究都发现，根据较大的差异度量触发警报会导致过多误报，尤其是在特征众多的数据集中。此外，大多数机器学习数据验证方法都会错误地针对预期的 异常触发警报。它们通过将最新的数据分区与其前一个分区或历史分区的单一聚合结果进行比较来验证数据 [8, 18, 51, 52, 58]。这种做法未能考虑到

*在 Meta 实习期间完成的工作

与时间模式相关的预期异常——例如，Prime Day 或黑色星期五等热门购物日可能会暂时推高用户参与度特征和广告点击率。更常见的是，用户参与度会随着典型的季节性规律而波动，例如昼夜或工作日与周末的模式。数据验证方法应针对意外的数据集损坏触发警报，也就是那些确实需要工程师关注的情况。

规模化的关键洞见：分区汇总。在查询优化和近似查询处理等其他数据管理场景中，人们采用过一种简单的思路，例如 [30, 36], : 存储各分区的摘要，以供后续计算使用。为了识别数据集损坏，我们可以将新分区的摘要与若干历史摘要进行比较，从而判断它是否是真实异常。例如，对于星期一对应的分区，比较对象不只是星期日的分区或上一周的粗粒度聚合结果，而是每周各天对应的单个分区——其中既包括周末，也包括工作日。我们将这种机器学习数据验证方法称为分区汇总（*PS*）。据我们所知，Redyuk 等人 [48] 提出的技术是此前机器学习数据验证领域中唯一采用分区汇总的方法。我们将其实现为基线，发现它仍会触发许多误报，因为它旨在对一般数据集错误发出警报，而非针对机器学习流水线错误；我们将在第 6.3.3 节进一步解释这一点。因此，我们的主要贡献是提出一个通用框架，用于将任意机器学习数据验证方法适配到 PS 场景。我们开发了几种适配方案，但由于它们未能满足我们的告警精确率需求，我们还创建了 gate，以便在我们的工作负载中实现高精确率。鉴于机器学习流水线数据集具有高度相关和高维的特点，gate 采用以下步骤：*(i)* 根据分区摘要对相关特征进行聚类，*(ii)* 计算特征的数据质量统计度量，以及*(iii)* 随时间监控集群范围聚合统计。

概述。本文概述了我们近期在 Meta 为机器学习流水线实现自动数据验证系统的工作。我们的系统专注于验证机器学习流水线的数据分区，以便在模型重新训练前检测问题。我们的贡献如下：

- 我们概述了机器学习流水线，并列举了自动数据验证系统的业务需求（第 2 节），
- 我们介绍了一种通用框架，用于将现有数据验证方法改造为分区汇总（PS）框架，但这些方法仍无法满足我们的需求——也就是说，它们仍会产生过多误报（第 3 和第 4 节），
- 我们介绍了 gate，这是一种采用 PS 框架的新型高精确率、高召回率自动数据验证技术（第 5 节），并且
- 我们讨论了 Instagram 机器学习流水线案例研究（第6节）的经验总结。我们将数据验证方法与 Redyuketal. [48] 进行比较，并讨论针对不同机器学习流水线场景的最佳方法。最后，我们举例说明如何在值班轮换期间使用我们的系统调试机器学习流水线的性能下降问题。

2 背景

本节介绍生产环境中机器学习流水线的背景、实用数据验证解决方案的需求，以及现有数据验证方法的分类。

2.1 机器学习流水线与模型验证

生产环境中的机器学习流水线由一个或多个机器学习模型组成，这些模型持续实时地产生预测，并定期使用最新的数据分区进行重新训练。训练与部署之间的界限并不清晰：数据处理管道会持续接收输入元组（即一行特征值），针对这些元组运行处于推理模式的机器学习模型（即不更改模型参数），随后再针对最新一批元组运行处于训练模式的机器学习模型（即更改模型参数）。

图1展示了一个生产环境机器学习流水线的示例。流向该流水线的流量会被路由至主模型模型 A，或数据科学家创建的多个实验模型之一（例如模型 B）。假设一个输入元组（① 图1中的输入元组）被路由至模型 A。那么，② 模型 A 将计算出一个预测（例如用户行为概率）。该预测会返回给终端用户或应用程序，并③ 记录到一张包含所有特征和预测的表中。另外，每隔几小时或几天（具体频率因模型和数据处理管道而异），④ 系统会启动重新训练任务，使用最新的元组分区重新训练每个模型，但会留出一小部分元组作为验证用的留出集。每个重新训练任务都会验证新训练的模型快照在留出集上的表现是否良好（即损失较低）；如果 ⑤ 这一步模型验证通过，新的模型快照就会取代对应的旧生产快照。机器学习工程师还可以加入针对特定模型的其他检查，例如对特定元组进行单元测试 [26]。

虽然模型验证对于防止糟糕的模型进入生产环境是必要的，但仅靠它还不够。Breck 等人 [8] 指出，代码中的错误是生产数据错误的常见来源；如果特征遭到破坏，机器学习预测的质量就会大幅下降。例如，移动应用的反馈日志记录机制可能发生故障，导致派生特征变为 null 或 -1，直到工程师修复该日志记录机制。在许多组织（如 Meta）中，此类错误的修复周转时间往往长于重新训练周期（也就是说，工程师需要花费几个小时以上来发现并修复错误，然后将变更推送到生产环境）——因此，模型可能会基于受损或异常的分布外数据 [54] 进行重新训练。由于留出验证集很可能也受到类似的污染，重新训练后的模型仍会通过验证阶段；然而，与历史性能相比，其在生产环境中的性能可能会大幅下降。因此，除了模型验证之外，还必须进行数据验证，尤其是在持续场景中，因为数据会不断反馈给模型以供重新训练。

2.2 Meta 的数据验证要求

我们的数据验证目标是确定训练数据集最新分区中是否存在足够多的“无效”元组，以致于如果将重新训练的模型快照推送到生产环境，会导致下游模型性能下降。显而易见，我们可以*gate*或阻止所有重新训练的模型升级到生产环境，但这样一来，

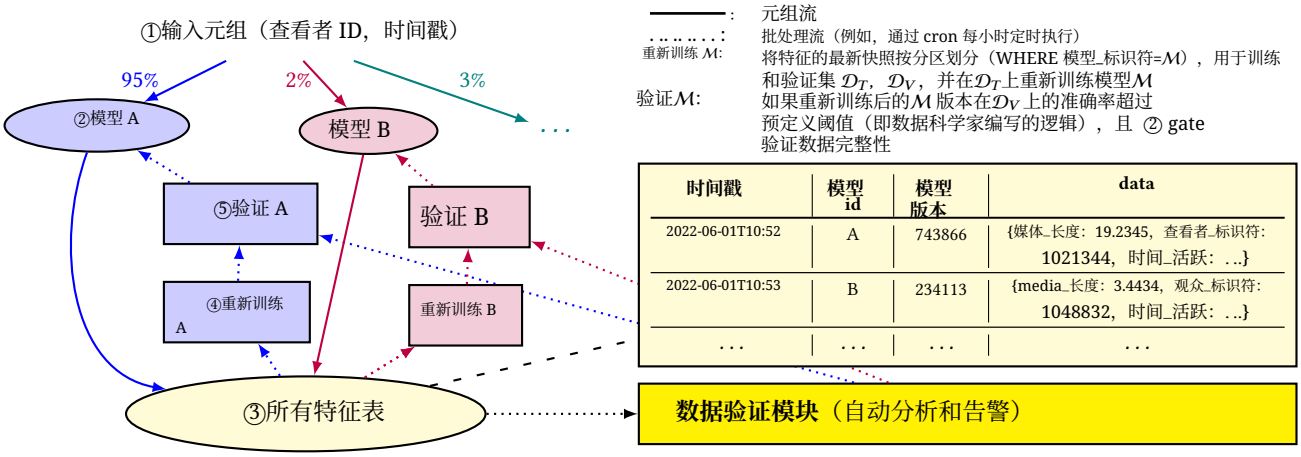


图1：用于单个预测任务的机器学习流水线，由一个主模型和许多实验模型组成。如果最新的特征快照已损坏，即使进行了针对模型的验证，生成的模型版本也会表现不佳。

生产环境中的模型快照很快就会过时。因此，*precisely* 识别数据何时以及如何遭到损坏至关重要。

尽管 Meta 的机器学习基础设施团队已经实现了 Breck 等人 [8] 和 Schelter 等人 [51], 所述的模式验证及其他约束检查技术，但这些技术无法精准捕捉我们的机器学习故障。此外，机器学习工程师必须为其数据处理管道枚举特征和约束；随着数据变化以及团队人员自然流动，要长期维持这一做法实在太困难。开始构建新的数据验证系统时，我们首先对 Meta 的机器学习工程师进行了非正式访谈，以收集需求，相关总结如下。

无需手动调优即可实现高精确率和高召回率的警报。工程师明确要求最低召回率（否则系统将毫无用处），并隐含要求最低精确率（否则他们不会关注系统警报）。因此，我们最初的方法鼓励工程师针对特征的完整性（即非空值所占的比例）设置警报。如果某项特征的完整性下降超过 30%，值班机器学习工程师就会收到通知。然而，这种方法产生了过多误报，机器学习工程师因此将其静音。一些机器学习工程师发现，手动调优阈值可以提高精确率，但这种做法需要付出大量努力。最终，机器学习工程师放弃了这种数据验证方法。**扩展到大量相关特征**。数据处理管道由数据集构成，其中可能包含数万个特征；许多特征高度相关，对整体机器学习模型性能的贡献也各不相同。若数据验证解决方案以相互独立且一视同仁的方式处理特征，就会触发误报。虽然一些组织或许能够根据特征重要性对特征的数据质量警报进行加权 [49], ，但这种方法无法通用，也无法在 Meta 的规模下实现计算上的可行性。并非所有模型都有明确的特征重要性分数（例如深度学习模型），而且模型经常重新训练，因此特征重要性也会频繁变化。**扩展到大量机器学习流水线**。Meta 的数据处理管道种类繁多，涵盖不同的应用程序（例如推荐、客户流失预测、自动回复）和终端用户（例如身处不同地理区域的客户）。机器学习流水线的数据摄取方式各不相同，训练动态也各不相同——例如，有些流水线遇到损坏数据的概率高于其他流水线，有些机器学习任务也比其他任务“更难”解决（例如，类别不平衡使欺诈检测颇具挑战性）。我们的数据验证平台必须能够支持各种流水线动态模式。

易于上手和使用。当机器学习团队提出数据验证需求时，我们必须立即推出解决方案，而不是花费数周或数月收集真实标签损坏的标注，再据此定制适用于其数据或数据处理管道的验证系统。此外，机器学习工程师通常要负责由他们并未创建的特征组成的机器学习流水线（由于组织人员流动）。要为他们可能并不了解的特征枚举并仔细调整约束和阈值，并不切实际。最后，警报应便于工程师理解并采取行动。由于工程缺陷通常只会破坏单个特征（例如，拼写错误会导致字符串型特征损坏），警报应对应到某个损坏的特征或一组特征 [55]。因此，监控难以理解的特征低秩表示（例如，主成分分析得出的表示）并无用处。

2.3 机器学习数据验证：现有度量指标

经典数据验证和修复技术关注的指标大致对应于正确性（例如，不违反模式 [45, 56]）、一致性（例如，遵循函数依赖）、完整性（即，保持较低的空值比例 [2, 48]）和统计属性（例如，相对于均值或中位数不存在离群值 [5, 28]）。数据验证方法 会采用一个或多个数据质量统计量，并在满足某个条件时触发警报。我们从三个维度对现有数据验证方法进行分类：

(1) 特征内：特征的点度量与密度度量 (2) 特征间：单个特征与特征组，或联合分布 (3) 时间：单个快照与随时间变化

随后，我们按照三个维度对更常见的数据验证方法进行分类，并展示如何通过简单扩展使任何方法都支持时间轴。

2.3.1 特征内验证。 用于评估单个特征分布的现有指标可分为“点”度量指标（如均值）和“密度”度量指标（如直方图或经验累积分布函数（eCDF））。以下统计量（涵盖点度量指标和密度度量指标）常用于数据清洗和验证：完整性、均值、标准差、唯一值数量、高频值数量，以及出现频率最高值的计数 [21, 51]。未能反映密度的指标可能会导致误报；例如，某个特征中的少数异常值可能会改变均值，却不会影响整体机器学习模型性能。

2.3.2 特征间验证。 大多数技术会独立且同等地监控单变量分布。在机器学习场景中，这可能导致误报，因为特征之间存在相关性，重要性也各不相同 [8, 33, 44, 51]。将这些技术扩展到多变量场景的一种方法，是将所有历史特征的数据质量度量拼接成一个向量，并使用基于模型的技术（例如， k近邻）将当前向量与历史向量进行比较 [6]。然而，如果不进行某种聚类，每个特征仍会被独立且同等地处理。一种解决方案是对特征分组，并计算每组的汇总统计量（例如，平均值） [8, 33]，但要求每位模型开发者或维护者手动对数千个特征分组并不现实。主成分分析（PCA）等降维技术可以降低数据的秩 [41]，或减少需要计算统计量的列数，但这些列无法解释——例如，如果发现某个主成分的均值异常，就难以判断该如何调试。此外，对拍字节级数据应用聚类技术在计算上不可行，因此需要进行一些抽样——但必须在样本中保留数据的时间特性，下一节将对此进行讨论。

2.3.3 时间验证。 特征的静态上下界会随着时间推移而过时，但我们可以跟踪滚动周窗口内的统计度量，以构建动态界限。假设我们每天计算统计度量 Q——那么在时间 t， $Q(t)$的有效范围可以是：

$$\frac{1}{7}\left(\sum_{i=1}^{7} Q(t-i)\right) \pm z \times \sigma\left([Q(t-1), Q(t-2), \ldots, Q(t-7)]\right)$$

其中， z 是标准差的阈值倍数， σ是标准差。基于机器学习的异常检测方法通常会考虑时间轴的存在，而无需进行扩展 [6]：例如，Redyuk 等人 [48] 使用最近邻模型比较不同时期的数据质量统计量。然而，这种方法不会根据时间对数据点赋予不同权重；例如，该算法无法区分昨天的数据质量统计量与一个月前某个任意日期的数据质量统计量。自回归模型通过跟踪统计量的估计移动平均值 [39]，为数据点赋予时间权重，但它们在处理多变量分布时效果不佳，尤其是在我们这种拥有数千乃至数万个特征的规模下。

展望未来。 要正确触发针对损坏数据分区的警报，数据验证方法必须覆盖三个维度——特征内、特征间和时间维度。许多现有数据验证方法在实现各维度目标方面的效果各不相同。例如，一种在这三个维度上表现都不佳的常见方法是计算某个特征的均值——比如照片的浏览量——并在浏览量超出围绕平均浏览量预先设定的静态范围时拒绝该元组 [8, 33, 51]。我们的分区摘要框架（详见第 4 节）同时采用点统计度量和基于密度的统计度量（特征内验证），并根据其移动平均值进行归一化（时间验证）。在第 5 节中，我们介绍了 gate 这一技术，它还通过以较低成本对特征进行聚类，并保留从簇映射回原始特征的映射（区别于主成分分析），来执行特征间验证。

3 问题

本节将介绍问题陈述，并讨论数据验证方法的评估指标。

3.1 形式化

考虑一个由按时间戳排序的分区组成的数据集 D[1] $\mathrm{D} = \{D^1, D^2 \ldots D^t\}$，其中 D 包含若干特征（即列）， $F_1, F_2, \ldots, F_n$， 而F_j^i表示数据第 i 个分区中特征 j 的值所构成的多重集，即 D^i。在每个时间戳 t，都会使用截至该时间点的分区重新训练机器学习模型，即 $\bigcup_{i<t} D^i$。尽管可以用准确率或损失等不同指标衡量模型的绝对性能，我们仍将机器学习性能下降 笼统定义为：相较于滚动基线，所选度量指标出现退化，例如准确率相较于 7 天滚动平均准确率下降 5%。通常，机器学习工程师会设定此定义和阈值。如果在数据分区 D^i 上重新训练的模型在下一个时间戳出现性能下降，即 $i+1$，则该数据分区 D^i 即为损坏 。我们用 $y^i \in \{0, 1\}$ 表示 D 中的损坏，其中 $y^i = 1$ 当且仅当 D^i 被损坏。

接下来，我们将数据验证方法 定义为一个函数：给定历史分区 $D^1, \ldots, D^{t-1}$，它会返回一个得分，表示 D^t 发生损坏的可能性。更正式地说，数据验证方法 f定义为 $f\left(D^t \mid D^1, D^2, \ldots, D^{t-1}\right)$。如果上下文已明确参数，我们就用简写 $f(t)$ 表示数据验证方法。实际上， $f(t)$ 的计算成本应该很低，尤其是在大规模场景下；但我们该如何评估其性能，也就是说，如何判断 $f(t) \approx y^t$? 接下来我们将讨论评估指标。

3.2 评估指标

通常，数据验证方法会产生警报，当 $f(t)$ 超过某个阈值（例如 50%）时，就会触发警报。给定警报阈值 τ，我们令 $a_\tau^t = \mathbb{I}\left[f(t) \geq \tau\right]$；因此我们希望 $a_\tau^t \approx y^t$，也就是说，警报应与数据质量的下降相符。我们在 t 处定义精确率和召回率：

$$P(t,\tau) = \frac{\sum_{i=1}^{t} y^i \cdot a_\tau^i}{\sum_{i=1}^{t} a_\tau^i} \quad R(t,\tau) = \frac{\sum_{i=1}^{t} y^i \cdot a_\tau^i}{\sum_{i=1}^{t} y^i}$$

数据验证方法最多可生成 t 个不同的分数，每个分数对应一个 $f(i)$，满足 $i \in \{1、\ldots\ldots、t\}$，因此最多有 t 个阈值 $\tau_1 \ldots \tau_t$ 可供选择。具体来说，在

[1]在本文中，每个分区代表一天（即 24 小时）的数据，但也可以采用其他划分方式（例如，每小时划分为一个分区）。

Meta，我们关注的是 **精确率@0.9**，也就是能达到 90% 召回率的阈值 τ 下的精确率。从业者主要关注这一度量指标，因为如果一种方法无法召回大多数故障，就达不到部署要求。除了精确率@0.9，我们还会考虑另一个度量指标——**平均精确率（AP）**——它综合考量了精确率-召回率（P-R）曲线在各个阈值 [47] 下的整体形状。AP 是各阈值下精确率的加权均值，权重为相较于前一个阈值的召回率增量。公式（1）给出了在 t 处 AP 的定义，其中 $P(t、\ \tau_i)$ 和 $R(t、\ \tau_i)$ 分别是第 i 个阈值 τ_i 下的精确率和召回率：

$$AP\left(t\right)=\sum_{i=1}^{t}\left(R\left(t,\tau_i\right)-R\left(t,\tau_{i-1}\right)\right)P\left(t,\tau_i\right) \tag{1}$$

平均精度（AP）反映了方法的整体预测能力，也就是在选择不同的分数阈值来触发警报时，该方法的表现。因此，平均精度为我们提供了对方法表现全面且无偏的估计。我们将在第7节讨论其他评估指标。

4 分区摘要

现有的数据验证设置 [8, 33, 51] 通常会对每个特征 F_j 在时间 t 应用某种统计度量 Q（例如，完整性、均值），并分析其 $Q\left(F_j^t\right)$ 与 $Q\left(F_j^{t-1}\right)$（即前一天的统计值）的比较结果。或者，它们会将 $Q\left(F_j^t\right)$ 与 $Q\left(\cup_{i=1}^{t-1}F_j^i\right)$ 进行比较，即在所有先前值的并集上计算出的统计值[2]。这两种设置都没有考虑预期的时间模式，例如周末与工作日表现不同。此外，后一种设置会丢失分区级粒度，而保留这种粒度对于识别节假日非常重要。

本文介绍的用于数据验证的分区汇总（PS）方法，将 Redyuk 等人 [48] 的方法扩展到任意度量，其步骤是先为每个分区 D^i 中的每个特征 F_j 计算一个或多个统计摘要（Q，例如均值），用于 $i \le t$，即 $Q\left(F_j^i\right)$。计算并按分区存储这些 Q 后，我们再以各种方式将它们组合起来，计算 f（t）。数据验证方法可能在分区摘要所用统计量的选择（即 Q）以及如何组合这些统计量以得出整体数据质量分数（即 f）方面有所不同。我们在第 4.1 节介绍如何将现有数据验证方法一般性地适配到 PS 场景，并在第4.2节给出具体的适配方法。最后，我们讨论在数据验证系统的初步版本中实现这些方法所获得的经验教训。表1列出了常用符号。

4.1 一般性适配

如第 3 节所述，在 PS 框架中，我们将数据验证分为两个步骤：首先，计算特征 F_1、F_2、……、F_n 在各分区 D^1、D^2、……、D^t 上的摘要 Q；其次，合并这些摘要，以获得每个时间步 t 的数据质量总体度量 $f(t)$。为了计算 $f(t)$，我们先汇总每个特征的差值，即该特征的值与其滚动平均值 Q 之间的差值。在实践中，我们需要先对 Q 和 Q 之间的差值进行归一化，然后再汇总。形式化地说，给定 Q 和 F_i，我们将最近 k 天的滚动平均值定义为 $\overline{Q}$：

$$\overline{Q}\left(F_i,t\right)=\frac{1}{k}\sum_{j=1}^{k}Q\left(F_i^{t-j}\right) \tag{2}$$

一旦得到每个特征的滚动平均值 Q，就可以按每个特征对这些 Q 进行归一化，得到 $\widetilde{Q}$。归一化后的 $\widetilde{Q}$ 可以使用不同的归一化技术计算，例如百分比差异（PD）或 z-得分：

$$\widetilde{Q}_{PD}\left(F_i,t\right)=\frac{Q\left(F_i^t\right)-\overline{Q}\left(F_i,t\right)}{\overline{Q}\left(F_i,t\right)} \tag{3}$$

$$\widetilde{Q}_z\left(F_i,t\right)=\frac{Q\left(F_i^t\right)-\overline{Q}\left(F_i,t\right)}{\sigma\left(\left[Q\left(F_i^{t-1}\right),\ldots,Q\left(F_i^{t-k}\right)\right]\right)} \tag{4}$$

其中，σ 是一个用于计算标准差的函数。最后，可以通过汇总每个特征的 Q 来计算 f，从而得到一个标量分数：

$$f\left(t\right)=\frac{1}{n}\sum_{i=1}^{n}\widetilde{Q}\left(F_i,t\right)$$

如果 $f(t)$ 超过阈值，我们就可以触发警报——阈值可以通过跟踪若干时间戳的 f 并判断当前值是否为异常值来确定。实际上，可以使用任何简单的标量异常值检测机制（例如，超过均值多个标准差，或高于第 95 百分位数）。由于我们的方法没有“训练集”，最初的几个时间步可能会错误地触发警报。不过，以这种方式选择的警报阈值能够抵御时间变化（也就是说，百分位数阈值会随时间推移保持其含义和重要性）。下文中，我们稍微放宽符号的用法，以便将我们的方法应用于各种数据质量指标：简而言之，Q 表示滚动平均值度量指标，Q 表示相对于其他时间步、使用 Q 对当前时间步进行归一化后的版本，而 $f(t)$ 则汇总各特征上的 $\widetilde{Q}$。

4.2 现有方法的改进

我们选取三类数据质量度量，将其改用于 PS 场景：（1）监控每个特征的完整性百分比降幅；（2）监控每个特征中任意标量统计量（例如完整性、均值）的 z-得分；（3）监控每个特征的双样本统计检验（例如柯尔莫哥洛夫–斯米尔诺夫 [34]）所得的 p 值。对于每种方法，我们在计算 Q 和 $\widetilde{Q}$ 时均设置 $k = 7$，即一周。我们选择 $k = 7$ 是出于组织运营方面的常见考量（例如，值班轮换通常以周为周期，会议也常按周举行），但这一参数可以调整。

4.2.1 δ-完整性下降。 完整性下降方法根据在时间 t 出现完整性下降的特征数量（即，$\widetilde{C}$），$\ge \delta$ 相对于其滚动 7 天平均完整性，来计算得分。该得分按每个特征的滚动平均完整性（即，C）加权。公式 (5) 展示了 −

[2]在此处以及其他地方，我们使用 ⊎ 时，指的是多重集并，而不是集合并。

符号	描述
D^t	时间戳 t处的数据分区
y	表示分区是否损坏的二元标签，$y^t = 1$ 如果 D^t 已损坏，则0 否则
F	数据集中的一个特征（即列）；F_j 表示第 j 个特征
f	数据验证方法，f: $D^t \to \mathbb{R}$
Q	数据质量统计量（例如均值、完整性），Q: $F^t \to \mathbb{R}$
$\overline{Q}$	某个特征的最新统计度量 Q 的均值，定义见公式 (2)
$\widetilde{Q}_{PD}$	Q 与其滚动平均值 Q之间的百分比差异，定义见公式（3）
$\widetilde{Q}_z$	z-得分差值（即标准差的数量），介于 Q 与其滚动平均值 Q之间，该值在公式（4）中定义
C	分区中特征的完整性（即非空值的比例），C: $F^t \to [0,1]$
G^t	GATE’s在时间 t 的聚类分配，其中 G^t（F）表示 F 簇，$\lvert G^t \rvert = v$ 是不同簇的数量

表1：符号表

时间 t的 δ-完整性下降得分：

$$\widetilde{C}_{PD}(F_i,t,\delta) = \begin{cases} \widetilde{C}_{PD}(F_i,t) \times \overline{C}(F_i,t) & \text{if } \widetilde{C}_{PD}(F_i,t) \geq \delta \\ 0 & \text{otherwise} \end{cases}$$

$$f_C(t,\delta) = \frac{1}{n}\sum_{i=1}^{n}\left|\widetilde{C}_{PD}(F_i,t,\delta)\right| \tag{5}$$

请注意，我们根据滚动平均值 C对每个超过 δ 的特征进行加权。这是因为，当某个特征相较于其历史滚动平均值发生较大变化且通常具有较高的完整性时，我们希望它能对得分 f_C 产生显著影响。时间 t 的得分 f_C 较高，意味着若干完整性较高的特征现在的完整度度量较低。

4.2.2 z-得分异常检测。在此方法中，我们固定一个标量统计量（例如完整性或均值），并根据在时间 t时具有超出某个截断值 τ的 z-得分的特征比例来生成得分。 z-得分使用该统计量的 7 天滚动均值和标准差计算。给定统计量 Q，时间 t 的异常检测得分如公式（6）所示：

$$\widetilde{Q}_z(F_i,t,\tau) = \begin{cases} \left|\widetilde{Q}_z(F_i,t)\right| & \text{if } \left|\widetilde{Q}_z(F_i,t)\right| \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

$$f_z(t,\tau) = \frac{1}{n}\sum_{i=1}^{n}\widetilde{Q}_z(F_i,t,\tau) \tag{6}$$

直观而言，对于 z-得分较低（即 $< \tau$）的特征，我们会将其贡献的 z-得分置零。这是因为，在必须监控的数万个特征中，许多特征很可能具有正常的 z-得分（即小于 3）。我们希望“最异常”的特征能够显著影响最终的 f_z 得分——例如，单个特征的z-得分为 15，比 15 个特征各自的 z-得分为 1 更值得警惕。在加入中间的 $\widetilde{Q}_z(F_i,\ t,\ \tau)$ 步骤（即 $\geq \tau$ 的指示函数）之前，这种方法的表现非常差。

4.2.3 双样本统计检验。为了考察每个特征的完整分布，我们基于三种不同的统计度量评估了 p-值：柯尔莫哥洛夫–斯米尔诺夫（KS）、Wasserstein-1（wass），也称为推土机距离，以及 DTS [17, 34, 59]。总体而言，KS 度量计算两个经验累积分布函数在单个取值处的最大差值；Wasserstein-1 度量计算两个经验累积分布函数之间差值的总面积；DTS 度量则根据合并后的经验累积分布函数的方差（在公式（9）中记为 $\hat{}D$）对 Wasserstein-1 度量加权。在我们的实现中，经验累积分布函数由 99 个百分位数组成，即从第 1 百分位数到第 99 百分位数。若**经验累积分布函数** $(F_i,\ t) \in \mathbb{R}^{99}$ 表示特征 F_i 在时间 t的经验累积分布函数，则双样本检验统计量如公式（7）至（9）所示：

$$\textbf{diff}(F_i,t) = \left|\textbf{eCDF}(F_i,t) - \frac{1}{7}\left(\sum_{j=1}^{7}\textbf{eCDF}(F_i,t-j)\right)\right|$$

$$h_{\text{KS}}(F_i,t) = \max \textbf{diff}(F_i,t) \tag{7}$$

$$h_{\text{wass}}(F_i,t) = \sum \textbf{diff}(F_i,t) \tag{8}$$

$$\hat{D}(F_i,t) = \sigma\left(\textbf{eCDF}(F_i,t), \frac{1}{7}\left(\sum_{j=1}^{7}\textbf{eCDF}(F_i,t-j)\right)\right)^2$$

$$h_{\text{DTS}}(F_i,t) = \frac{\sum \textbf{diff}(F_i,t)}{\hat{D}(F_i,t) \times \left(1 - \hat{D}(F_i,t)\right)} \tag{9}$$

请注意，**差值** $(F_i,\ t) \in \mathbb{R}^{99}$ 表示时间 t 的经验累积分布函数与 7 天滚动平均经验累积分布函数之间的逐元素绝对差。为了将统计度量转换为 p-值，我们通过随机划分当前和历史平均经验累积分布函数，对该统计度量进行自助法估计，并计算自助估计值未超过原始度量的比例。

为了将 p-值转换为 t 的标量分数，我们计算特征中 p-值小于显著性水平 α 的比例，该水平通常设为 0.05。如果 $p(F_i,\ t)$ 是在时间 t 对特征 F_i 进行双样本统计检验所得的 p-值，则该检验在时间 t 的得分如公式（10）所示：

$$\widetilde{Q}_{\text{twosamp}}(F_i,t,\alpha) = \mathbb{I}\left[p(F_i,t) \leq \alpha\right]$$

$$f_{\text{twosamp}}(t) = \frac{1}{n}\sum_{1=1}^{n}\widetilde{Q}_{\text{twosamp}}(F_i,t,\alpha) \tag{10}$$

4.3 经验教训

尽管上述对现有数据验证技术的改进确有益处（正如我们将在第 6.3.1 节中看到的），但在与 Meta 的机器学习工程师开展初步合作时，这些方法未能满足我们的精确率@0.9 需求。本文将讨论各方法的主要启示。

4.3.1 δ-完整性下降。 请记住， δ 表示特征的完整性从滚动平均值下降多少时会触发警报的阈值。为求简便，Meta 的工程师倾向于为每个特征设置相同的 δ，导致出现许多误报。此外，工程师花了几个月时间调节 δ，

我们发现，理想的 δ 在不同数据集之间差异很大，因此需要一种无需如此多手动调参的方法。

4.3.2 *z*-得分异常检测。 请回忆第 4.2.2 节的内容：该方法适用于任何标量统计量 Q（例如均值、完整性）。我们发现，完整性在精确率@0.9 和平均精度方面表现最佳；我们推测，这是因为与其他统计量相比，完整性产生的假阳性更少（例如，大量空值这一意外情况，比均值较大更可能损害下游机器学习模型性能，而均值较大可能是由少量离群值造成的）。与完整性下降方法类似，在要求高召回率时，这种得分异常检测方法仍会产生许多假阳性。

4.3.3 双样本统计检验。 尽管文献似乎表明，KS 检验是一种常用的机器学习漂移检测实用技术 [16, 32, 44], ，但我们发现，KS 检验也会产生大量假阳性。深入分析后发现，KS 检验实际上不适用于我们的问题。如第 4.2.3 节所述，KS 检验只考虑两个经验累积分布函数之间的最大距离（即 L^∞）。这在机器学习场景中并不理想，以下情形便说明了这一点：设想两个经验累积分布函数（当前和历史）基本相同，只是当前经验累积分布函数中有 1% 的元组是离群值。由于只有一小部分元组受到污染，下游机器学习模型性能可能不会明显下降，但 KS 检验统计量却可能很大！因此，我们发现，基于经验累积分布函数间 L^1 距离的统计检验（例如 Wasserstein-1、DTS）表现远优于 KS 检验，Breck 等人的实验结果也证实了这一点 [8]。Wasserstein-1 的表现最佳。此外，由于需要通过自助法生成 p-值，双样本检验的运行速度可能比其他方法慢几个数量级。我们发现，这类方法无法扩展到我们的高维数据集，其中一些数据集包含数十万列。

5 GATE

本节介绍 gate，即我们提出的技术。与第 4.1 节中的方法相比，该技术能减少假阳性，但实现起来稍微复杂一些。

5.1 现有 PS 改编方法的缺点

在设计 gate（图 2）时，我们发现了第 4.3 节中所述的现成改编方法存在若干缺点。首先，这些方法将各个特征视为对机器学习预测具有同等且独立的重要性。正如第 1 节所讨论的，这并不适用于我们的机器学习流水线。因此，聚类对于减少高维数据集中误报的警报至关重要，但要大规模进行聚类并映射回原始特征以便调试并不容易。我们采用的解决方案是，直接对分区摘要的协方差矩阵应用谱聚类，这种方法运行速度快，而且能保留原始特征。其次，以往的方法无法捕捉特征分布的完整密度，而像均值这样的“点”统计量通常会遗漏这一点。为此，我们在 gate 中的摘要包含多种统计量，其中包括不同时间戳之间累积分布函数的差值。第三，很难确保统计量能够适应预期的时间变化并保持稳健：例如，单调递增的特征（如总数量）会随时间变化，但我们不希望将其标记为错误。为此，在进行比较之前，我们会根据近期分区摘要对统计量进行归一化，将其转换为 *z*-分数。

5.2 gate 的成分

为了获得较高的精确率和召回率，我们的技术 gate 可分解为以下几个成分（如图 2 所示）：

(1) **去相关：** 为了解决特征间验证问题，我们对特征之间的相关性应用谱分解。去相关步骤接收输入分区 D^i，其中 $i \le t$，并输出 $G^t \in \mathbb{R}^n$，其中每个特征仅由一个簇表示。我们将唯一簇的数量记为 $|G^t| = v$。为了提高速度和可扩展性，我们将这种聚类方法应用于概要或汇总统计量，而非整个底层数据集。（第 5.2.1 节）

（2）**异常矩阵创建：**为了解决特征内验证问题，我们为每个特征计算 6 个统计度量，并计算其 z 得分，以便进行归一化。我们使用 $G^t \in \mathbb{R}^n$ 对 z得分进行聚类并求平均值。此步骤返回 $X\ (t) \in \mathbb{R}^{t\times 6v}$，其中 $X\ (t)$ 中的每一行都表示该时间戳的数据质量度量。（第 5.2.2 节）（3）**警报生成：**为了解决时间验证问题，我们计算 $X_t\ (t)$（即 X中的第 t 行）与一部分最接近的前序行之间的平均距离；如果平均距离超过某个阈值，就会触发警报。也可以根据需要筛选近期时间戳。此警报生成步骤接收上一步的 $X\ (t) \in \mathbb{R}^{t\times 6v}$，或前序分区拼接后的异常矩阵，并输出损坏分数。（第 5.2.3 节）（4）**（可选）特征下钻：**虽然此步骤对告警机制并非至关重要，但机器学习工程师希望了解触发警报的原因。我们会找出最异常的簇，并深入分析其中的特征及对应的异常统计量。（第 5.2.4 节）

接下来，我们将更详细地讨论这些步骤 .

5.2.1 去相关。 由于数据集可能包含数万个特征，其中许多彼此相关，有些则对下游机器学习模型性能的贡献并不显著，因此在实际应用中，独立监控各个特征并赋予相同权重，很可能会触发误报。为了降低误报的可能性，我们发现，将特征聚类并监控由此形成的各组汇总信息，而非单个特征，至关重要。为避免数据泄漏，我们使用约一个月的历史数据进行聚类，这样方法就不会通过查看未来数据来识别特征之间的相关性。每个数据分区可能有数拍字节，因此对原始数据进行聚类并不现实。因此，在这一步聚类中，我们聚类的是分区摘要中的信息，而不是原始数据本身 [65]。

为了进行聚类，我们采用了图论方法：节点是特征，边距离对应于端点节点（即特征）之间的相关性 [60]。首先，对于 6 个汇总统计量中的每一个（详见第 5.2.2 节），我们使用汇总表计算特征间的协方差矩阵。为了确定簇的数量 $|G^t|$，我们对每个协方差矩阵运行主成分分析，以确定能够解释 95% 方差的成分数量 v_1，……，v_6。我们设定

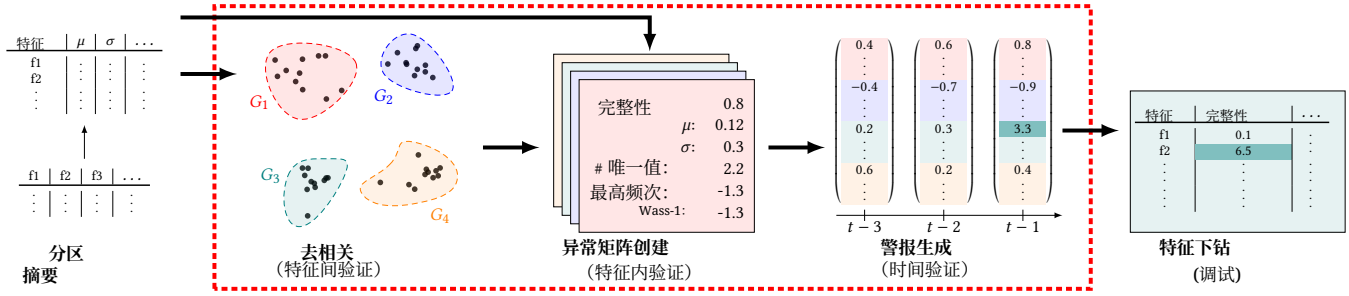


图 2：我们用于机器学习数据验证的分区汇总（PS）方法架构。红色方框表示一种通用的数据验证技术。我们发现，现有技术在识别机器学习性能下降方面的精确率和召回率都不高——主要问题是误报太多。我们的方法GATE会对相关特征进行聚类，并在整个组的汇总统计量出现异常时触发警报。

$|G^t| = \max v_1, \ldots\ldots, v_6$。然后，我们对协方差矩阵的绝对值求和，得到图边矩阵。最后，我们运行谱聚类算法，将图划分为聚类分配 G^t，其中每个特征对应一个簇。我们将特征 F所属的簇记为 $G^t(F)$。

文献中有多种聚类方法 [63]；在这里，我们选择了谱聚类，因为它简单且适用于图结构。由于我们想对特征（即列）而非元组进行聚类，因此只需构建一个以特征为节点、以相关性为边权重的图。随后，应用谱聚类算法即可根据特征之间的相关性识别相似特征组。谱聚类非常适合非凸数据集，但也有一些缺点：它无法处理噪声数据，且运行复杂度较高 [22]。不过，我们的方法不会遇到这些问题，因为我们的摘要是聚合结果，因此不太可能含有噪声；并且协方差矩阵（即图）的大小取决于特征数量，而非数据集中的元组数量。图的大小也不取决于分区数量，因此聚类运行时间会始终保持不变。

5.2.2 异常矩阵创建。此步骤的目标是将 D^t转换为统计量向量，以便与先前分区的向量进行比较——本质上是创建一个统计量矩阵，用来表示所有 D^1、D^2、……、D^t。我们的算法使用了以下六种统计量：(1) 完整性，(2) 均值，(3) 标准差，(4) 唯一值的数量（通过草图近似），(5) 最高频次（即出现频率最高的值的计数除以总计数），以及 (6) 连续分区的经验累积分布函数之间的 Wasserstein-1（即 Earth-Mover's）距离。为对不同特征和时间的统计量进行归一化，我们将每种统计量转换为一个 z-得分。然后，我们通过对各簇的 z-得分取平均来降低维度。聚类步骤（第 5.2.1 节）基于相关性对特征赋予不同权重，以用于警报生成步骤（第 5.2.3 节）。

给定第 5.2.1 节中的聚类分配 G^t、$|G^t| = v$ 个不同的簇，以及时间 t 的统计量 Q_1，Q_2，……，Q_6，我们对簇内的归一化统计量 $\widetilde{Q}_z$ 取平均，得到公式（11）：

$$\begin{aligned}
\widetilde{\mathbf{Q}}_z(F_i,t) &= \left[\left|\widetilde{Q1}_z(F_i,t)\right|,\ldots,\left|\widetilde{Q6}_z(F_i,t)\right|\right] && \in \mathbb{R}_+^6\\
\widetilde{\mathbf{Q}}_G(j,t) &= \frac{\sum_{i=1}^n \mathbb{I}\left[G^t(F_i)=j\right]\cdot\widetilde{\mathbf{Q}}_z(F_i,t)}{\sum_{i=1}^n \mathbb{I}\left[G^t(F_i)=j\right]} && \in \mathbb{R}_+^6\\
\mathbf{x}(t) &= \left[\widetilde{\mathbf{Q}}_G(1,t)\ \widetilde{\mathbf{Q}}_G(2,t)\ \ldots\ \widetilde{\mathbf{Q}}_G(v-1,t)\ \widetilde{\mathbf{Q}}_G(v,t)\right] && \in \mathbb{R}_+^{6v}
\end{aligned} \tag{11}$$

请注意，$\mathbb{I}\left[G^t(F)=j\right]$是一个二元函数：如果 F属于第 j 个簇，则返回 1，否则返回 0，具体取值由 G^t决定。现在，我们计算 $X(t)$，这是一个表示当前和历史分区中$\mathbf{x}$ 的矩阵。 $X(t)$ 中的每一行对应一个分区，各列则表示拼接后的归一化统计向量（涵盖不同特征和时间）。

$$X(t) = [\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(t-1), \mathbf{x}(t)] \qquad \in \mathbb{R}_+^{t\times 6v} \tag{12}$$

请注意，必须对每个分区应用相同的聚类，即 G^t，否则比较 $X(t)$ 中的行就没有意义。

5.2.3 警报生成。 根据上一步得到的 $X(t) \in \mathbb{R}^{t\times 6v}$，比较各行，以确定时间 t 的数据是否应触发警报。直观地说，每行，即 $X_i(t)$，代表 D^i 的质量。我们设定一个邻居比例 f，并针对每一行$i \in t$，根据公式（13）计算 $X_t(t)$ 到最近的$\lfloor f \times t \rfloor$ 个非异常邻居（即前序行）的平均距离：

$$\begin{aligned}
\mathbb{I}_y(t) &= \begin{cases} 1 & \text{if } y^t = 1 \\ \infty & \text{otherwise} \end{cases}\\
\mathbf{d}(t) &= \left[\mathbb{I}_y(i)\cdot\|X_t(t)-X_i(t)\|_p \text{ for } i\in[1,\ldots,t-1]\right] && \in \mathbb{R}_+^{t-1}\\
f_{\text{GATE}}(t) &= \frac{1}{\lfloor f\times t\rfloor}\sum \min_{\lfloor f\times t\rfloor}\mathbf{d}(t) && \in \mathbb{R}_+
\end{aligned} \tag{13}$$

我们尝试在公式 (13) 中为 p 设置 $p = 1$ （曼哈顿距离）和 $p = 2$ （欧几里得距离）。与第 4.1 节所述的改编方法类似，如果$f_{\text{GATE}}(t)$ 超过阈值，我们就会触发警报；该阈值可通过持续跟踪

几个时间戳内的 f_{GATE}，并判断 f_{GATE} 的当前值是否为异常值来确定。我们的警报阈值具有可解释性：将“允许的最大值” z 得分乘以簇的数量（即 v），即可估算 X^t 中两行之间的距离可以有多近。

5.2.4 调试：特征下钻。 尽管下钻组件并非我们自动化数据验证方法中的一个步骤，gate 的设计目的是帮助从业者通过深入查看相关簇的特征来调试警报。首先，我们根据各簇对应的 $\tilde{\mathbf{Q}}_G$ 在 $\mathbf{x}$ 中的值对簇进行排序（公式（11））。然后，对于排名靠前的簇，我们使用 G^t 识别每个簇中的特征，并以类似方式根据其 $\tilde{\mathbf{Q}}_z$的幅度对特征进行排序。根据 z-得分的定义，排名靠前的特征是行为最异常的特征（也就是说，绝对值更大的 z-得分距离均值的标准差数更多），可以呈现给机器学习工程师。另一种有用的调试策略是将 $X_t(t)$ 与异常分区进行比较，或者在 $i < t$ 中将 $X_i(t)$ 与 $y^i = 1$进行比较。如果 $X_t(t)$ 是季节性异常，例如感恩节和圣诞节，那么相邻的异常 $X_i(t)$也可能对应某个节假日，因为每个分区都是独立汇总的。可以在第 5.2.3 节所述的警报中添加更多信息，以提供有关节假日及其他实用模式（例如工作日与周末模式）的背景信息。我们在第 6.3.2 节举例说明下钻组件。

6 案例研究

本节中，我们报告了对 Instagram 两条机器学习流水线进行案例研究的结果。首先，我们给出第 4 节和第 5 节所述 PS 方法的精确率@0.9 和平均精度数值，以及它们的运行时间。我们根据业务考量选择参数值（例如，针对数据的滚动 7 天聚合计算统计量）。随后，我们讨论这些技术的适用性——它们适合哪些类型的机器学习流水线，以及在值班轮换期间可能引发的告警疲劳。

6.1 设置

6.1.1 数据集描述。案例研究中的两个数据集都包含一个月的分区（即 D^1，D^2、……、D^t其中 $t \approx 30$）以及数万个特征（即 F_1，F_2、……、F_n其中 $n \geq 20,000$）。我们将这两个具体数据集分别记为 D_1 和 D_2，其特征集分别为 F_1 和 F_2。由于本案例研究是在第一作者实习期间开展的，这些数据集取自夏季。D_1的最早日期分区为 2022 年 6 月 5 日，最晚日期分区为 2022 年 7 月 5 日。D_2的最早日期分区为 2022 年 7 月 15 日 15 日，最晚日期分区为 8 月。数据集 D_1 和 D_2 共享部分（但并非全部）特征——具体而言，$F_1 \neq F_2$ 和 $F_1 \cap F_2 \neq \emptyset$。出于隐私和法律原因，我们无法披露特征详情，但数据集包含整数、浮点数和类别特征。

如第 2.1 节所述，机器学习流水线中的模型会频繁地自动使用 D 的最新视图进行训练，并串联起来生成最终的浮点数预测，表示用户点击媒体推荐内容的概率（二元分类）。我们直接与 Instagram 值班工程师合作，确定他们希望数据验证系统标记哪些类型的机器学习性能下降。我们发现了三类：

(1) 模型损失增加：针对某个机器学习模型（使用 D 训练），实时预测的（归一化）交叉熵损失是否比其滚动 7 日平均交叉熵损失高出某个预设阈值 **(2) 未经校准的预测：** 针对校准函数（基于机器学习预测拟合，用于使预测与真实事件相符），其误差是否大于某个预设阈值 **(3) 标签偏移：** 用户信息流中的媒体点击次数是否比其滚动 7 日平均点击率低了某个预设阈值

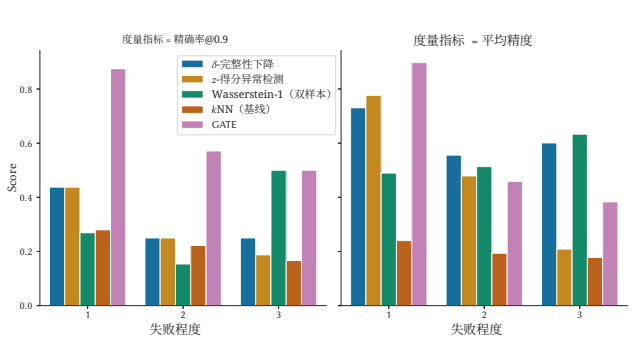
对于每个 D，我们将三种故障事件的所有出现情况合并，得到数据验证方法的真实标签 y^1，……，y^t。如表 2 所示，D_1 和 D_2 的故障率（即正例所占比例）显著不同。D_2 的故障率异常高，因此方法更容易取得良好的精确率和召回率。

最后，正如第 2 节所述，我们组织中的机器学习团队看重数据验证方法的可调节性，也就是根据不同程度的故障（即机器学习流水线的模型性能下降）调整方法的能力。例如，发生在重大国际新闻公告发布日的故障，可能比一年中“平常”日子的故障影响更大。为了在案例研究中纳入可调节性，我们为每个数据集或机器学习流水线设定了不同的故障级别。为确定单一的故障级别（FLs），我们按照表 2 所示，为每种故障定义设定了阈值。我们设定了三个严重程度逐渐增加的故障级别（FL_1，FL_2，FL_3）。各项定义的阈值逐步提高，使得 FL_i 中的故障数量为 ≥ 中的故障数量的 FL_{i+1}。机器学习工程师对这些故障级别进行了人工验证，以确保正向真实标签对应的是工程师需要修复的实际模型性能问题。

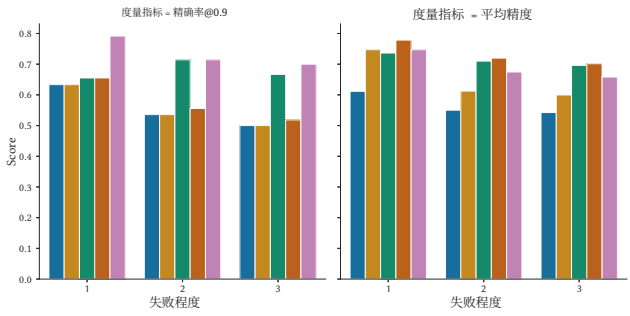
FL	定义	$\mathcal{D}_1$	$\mathcal{D}_2$
1	(↑ 交叉熵 ≥ 0.02) ∪ (校准误差 ≥ 0.3) ∪ (标签偏移 ≥ 0.3)	0.33	0.59
2	(↑ 交叉熵 ≥ 0.04) ∪ (校准误差 ≥ 0.4) ∪ (标签偏移 ≥ 0.4)	0.23	0.47
3	(↑ 交叉熵 ≥ 0.06) ∪ (校准误差 ≥ 0。5) ∪ (标签偏移 ≥ 0.5)	0.17	0.44

表2：$\mathcal{D}_1$ 和 $\mathcal{D}_2$中每个故障级别（FL）的正标签（即故障）占比。

6.1.2 数据验证方法。 我们将 Redyuk 等人 [48]作为基线，与第4节中描述的改进方法进行比较（δ完整性下降、 z得分异常检测、Wasserstein-1 双样本统计检验），并与 gate 进行比较。Redyuk 等人 [48] 提出的方法会为每个时间步创建一个统计量向量，并使用 k近邻算法将该向量与历史向量进行比较，从而将当前时间步的向量标记为异常或可接受。我们使用论文中的 7 个统计量：完整性、不同值的近似计数、最频繁值的比例、最大值、均值、最小值和标准差。对于每个特征，我们计算这些统计量，并将结果连接起来，得到用于 k近邻算法的更大向量。我们在图3中将此方法称为 kNN（基线）。



(a) 只有少数故障时的结果 $\mathcal{D}_1$，我们观察到的大多数机器学习流水线也有类似的低故障数。



(b) $\mathcal{D}_2$，存在大量故障（> 50%）时的结果。我们的方法实现了更高的精确率@0.9，并取得了与基线相当的平均精度[48]。

图3：Instagram 案例研究在不同数据集和故障级别上的结果。图例见图3a。

6.1.3 架构设置。 出于隐私和法律方面的考虑，我们仅讨论架构设置的高层细节。

每个数据集的原始数据（即 D）都存储在数据仓库系统中。新的分区到达后，会启动一个 PS作业：该作业会计算该分区的机器学习模型性能（例如准确率或损失）和汇总统计量，并将结果记录到汇总存储中。该作业的查询使用 PrestoSQL 编写 [53]。作业中计算的统计量涵盖数据验证方法所需的所有统计量，例如每个特征的均值、标准差、最小值、最大值、完整性、经验累积分布函数、高频值等。因此，每种数据验证方法都会从同一汇总存储中读取数据，以比较摘要并触发警报。这些数据验证方法使用 Python、Pandas 和 scikit-learn 实现。每个分区摘要都会运行这些作业，并使用 scikit-learn 函数计算评估指标平均精度和精确率@0.9。

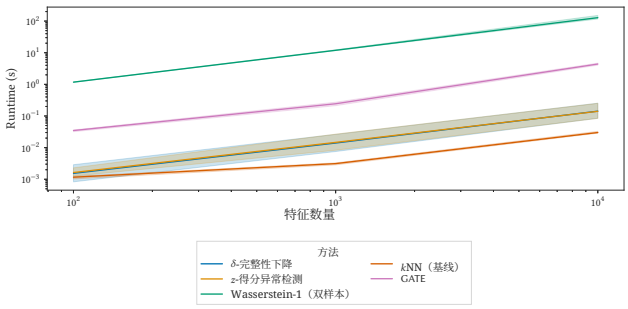


图 4：使用三次试验，计算不同特征数量下单个分区得分的运行时间。 δ- 完成度下降和 z-得分异常检测方法的运行时间相近。

6.2 性能结果

对于 D_1，我们表现最佳的方法使精确率@0.9平均提升了 2.9×，AP 平均提升了 2.3×，相较于基线有所改善。对于故障更多的 D_2，我们表现最佳的方法使精确率 @0.9 提升了 1.3×，同时 AP 保持与基线相同。我们观察到其运行时间较为合理，约为 5 秒（若加上第 5.2.1 节所述的离线去相关步骤，则为 50 秒）。图3展示了第6.1节所列各方法的结果。图 4展示了各方法计算单个分区得分所需的运行时间，并对特征数量进行了变化。我们将在第 6.3.1 节讨论如何根据不同的机器学习流水线设置选择方法。

6.2.1 精确率*@0.9*。在 D_1 中，gate 的精确率@0.9 显著优于其他方法。对于 D_1 中的故障级别 1 和 2，gate 的精确率@0.9 约为其他方法的 2×。对于故障级别 3，Wasserstein-1 双样本检验的精确率@0.9 与 gate 持平。在 D_2 中，gate 在所有三个故障级别上都取得了最高的精确率@0.9。对于故障级别 2，Wasserstein-1方法的表现与 gate 持平；对于故障级别 3，表现也较为接近。在 D_2 中，故障更多，gate 在精确率@0.9 方面的提升并不那么显著；尽管如此，在所有故障级别中，gate 的精确率仍然最高。gate 的精确率较高，得益于其聚类步骤，该步骤可防止相关特征触发大量误报。故障较少时（即故障级别较高时），Wasserstein-1 双样本检验的精确率较高，因为发生严重故障时，大量记录会受到破坏，从而显著扰动整个特征分布。与点统计量（如均值）相比，密度度量（如 Wasserstein-1 距离）能够更准确地捕捉这种分布偏移，而且不易受异常值影响。

6.2.2 平均精确率（*AP*）。虽然对我们而言，AP 不如精确率@0.9 这一度量指标重要，但我们发现，在 D_1 中，我们的方法的 AP 高于基线，在 D_2中则几乎与基线持平。在 D_2中，故障数量显著更多，Redyuk 等人 [48] 提出的基线在所有故障级别（FL）上均取得最高 AP，但 gate 达到了该值的 90%，而 Wasserstein-1 双样本检验达到了该值的 97%。尽管如此，我们仍认为，在这种情况下，我们的方法取得的 AP 是可接受的，因为大多数机器学习流水线不会经历如此高的故障率（此时更容易达到较高的精确率）。

6.2.3 运行时间。 基线（*k*NN）是速度最快的方法，因为无需计算基于时间的归一化。双样本

检验（平均精度最高）计算开销最大，因为它们需要通过自助法得出 p 值。对于超过 $O(10,000)$ 个特征的情况，这些检验并不适合执行。gate 的运行时间排名第二，但处理 $O(10,000)$ 个特征时仅需几秒。虽然 gate 包含 Wasserstein-1 距离度量，但它比双样本检验更快，因为它无需通过自助法计算 p 值。

6.3 讨论

在此，我们讨论不同机器学习流水线设置下应选择的方法、gate 的特征下钻组件，以及 Redyuk 等人基线方法的局限性 [48]。

6.3.1 不同设置下的最佳 *PS* 方法。 在 Instagram 机器学习流水线案例研究、与工程师合作的经历，以及与其他团队共同为数据验证系统制作原型的过程中，我们发现，如果数据验证系统需要达到较高的精确率@0.9，不同设置下的最佳 PS 方法也各不相同：

低故障率。 当数据处理管道中的数据经常损坏时，根据定义，这些损坏就不那么异常，因此像 z-得分计算这样的异常检测技术便毫无用处。单独使用 Wasserstein-1 检验（即捕捉特征的分布）是最佳的单一度量指标，但计算成本高昂，因为需要通过自助法获得 a p-值。gate 的精确率更高，包含 Wasserstein-1 距离，且计算成本低了好几个数量级（图 4）。我们的大多数机器学习流水线故障率相对较低，因此 gate 是一个不错的通用选择。然而，对于故障率较高的流水线，尤其是在平均精度比精确率@0.9更重要时，Redyuk 等人 [48]的方法或许更合适。

许多相关特征。 当机器学习流水线中有许多相关特征时，gate 在减少假阳性方面显著优于其他方法，这得益于 gate 的聚类组件（第 5.2.1 节）：只有当一整组相关特征都呈现异常时，才会触发警报。在与其他团队合作时，我们观察到，许多机器学习流水线有数百个、甚至数千个相关特征。与许多其他组织的情况一样， [54], 数据科学家经常根据会话数据创建相关特征——例如，可能会有一些二元特征，分别表示视频播放超过 1 秒、2 秒、5 秒等。不过，对于特征较少的数据处理管道，像 δ-完整性下降这样简单、快速的方法可能更合适。

部署的第一周。 我们注意到，在部署初期，最近邻方法生成警报的方式似乎是随机的。这是因为没有足够的数据可供比较。由于大多数数据处理管道的故障率通常小于或等于 1/7，因此至少需要一周，gate 才能开始生成有意义的警报。在第一周，其他方法（例如， δ-完整性下降）可能更适合用于触发警报，因为警报标准通常更有意义（例如，下降 30%、 z-得分 ≥ 3、 p-值 < 0.05）。

高基数特征。 当特征的取值分布范围很大时，gate 效果很好。对于只有二元特征或低基数特征（即仅涵盖少数几个不同取值）的场景，前 k 个值或唯一值数量等数据质量指标可能仅凭自身就能准确标记异常。然而，对于实值或其他高基数特征，即使特征取值的分布保持相似，这些指标也可能出现波动。实际上，许多机器学习流水线中都包含相当数量的高基数特征：例如，用户和内容嵌入通常是浮点值，并常用于机器学习流水线 [12, 40]。

6.3.2 值班体验。 在这项案例研究中，gate 通过减少误报，在满足可接受的召回率要求的同时，降低了值班轮换期间的告警疲劳。尽管 gate 主要用于在机器学习性能下降出现在性能仪表板上之前进行预测，但特征下钻组件（第 5.2.4 节）也可用于值班轮换期间，以诊断性能下降的原因。在一个场景中，某个机器学习流水线的性能显著下降，几天过去了，仍未找到根本原因。由于该模型包含超过 10,000 个特征，大多数特征至少有一个异常的数据质量统计量，因此无法找出出问题的特征组。查看 gate 识别出的最异常簇中的特征名称后，很明显这是一个声音相关缺陷，因为许多特征都与音频有关，且其 Wasserstein-1 值异常。

6.3.3 基线方法的局限性。 虽然 Redyuk 等人 [48] 提出的通用数据质量方法在故障率较高的机器学习流水线场景中取得了良好的平均精度，但我们发现，其精确率@0.9 太低，无法在实际环境中应用。首先，该方法不适用于时间序列场景，因为其归一化步骤（即计算 $\widetilde{Q}$）不会根据时间对分区摘要进行归一化。为了解这可能带来的问题，请考虑两个分区的摘要 $\mathbf{h}(1)$ 和 $\mathbf{h}(2)$，其中包含两个统计量（完整性和均值）以及三个特征：

$$\mathbf{h}(1) = [0.9, 13, 0.4, 1993, 0.8, 284448]$$
$$\mathbf{h}(2) = [0.9, 13, 0.4, 1993, 0.2, 300]$$

每个$\mathbf{h}$ 都是每个特征对应的（空值比例，均值）元组的串联，而$\mathbf{h}(1)$和$\mathbf{h}(2)$之间唯一的差异是第三个特征的空值比例和均值。如果我们按照 Redyuk 等人 [48], 的方法进行归一化，或将每个向量重新缩放到 0 到 1 之间，就会得到：

$$\mathbf{x}(1) = \left[1.8 \times 10^{-5}, 4.4 \times 10^{-4}, 0., 7.0 \times 10^{-2}, 1.4 \times 10^{-5}, 1.\right]$$
$$\mathbf{x}(2) = \left[3.5 \times 10^{-4}, 6.4 \times 10^{-4}, 1.0 \times 10^{-4}, 1., 0., 1.5 \times 10^{-1}\right]$$

归一化后，$t = 1$ 到 $t = 2$ 之间保持不变的统计量如今相差数个数量级——尽管只有一个特征发生了变化。这种差值会使告警触发机制产生混淆，或让工程师将注意力错误地放在其他特征上（例如，实际上并未发生变化的特征）。在错误类型较少或列数较少的数据集中，归一化中的这一细微缺陷可能不会引起注意；但在实践中，机器学习流水线的数据集可能包含数千个甚至更多特征（即列），而且几乎每个分区中总会有另一个特征带有一个或多个异常值。因此，图3中的结果显示，这种方法的精确率较低。gate 使用 z-分数，按时间对统计度量进行归一化，从而避免了这个问题。我们观察到的另一个问题是，Redyuk 中的 kNN

等人 [48] 对每个特征一视同仁，这在机器学习场景中并不实用，因为此类场景中的数据处理管道往往包含许多相关特征，且特征重要性各不相同。

7 相关工作

gate 综合了异常检测、聚类、数据清洗和数据验证方面的见解。

面向大型数据集的异常检测与聚类。 Chan- dola 等人 [10] 将异常检测定义为在数据中寻找不符合预期行为的模式，并列举了随着数据的底层分布随时间变化，如何调整异常概念所面临的挑战。虽然基于阈值的估计模型（例如 z-得分、中位数绝对偏差）通常用于估计单点离群值 [6, 35], , 但我们的问题更接近于寻找离群值子序列这一挑战。我们从不相似性方法（例如 k-近邻）中汲取灵感，这类方法通常用于估计离群值子序列 [3, 6, 46], ; 但在我们的场景中，挑战在于恰当地归一化输入并处理相关特征，详见第 5 节。多篇论文探讨了高维数据 [23, 25, 50]中的异常检测。像主成分分析 (PCA) 这样的方法可以生成数据的低秩表示，从而产生可在元组级别而非特征或列级别上解读的警报（这是工程师提出的要求，详见第 2.3 节）。Blázquez-García 等人 [6] 综述了数据子序列的聚类技术，但这类方法无法适用于 Meta 的规模。Zhang 等人 [65] 介绍了使用数据分区对大型数据集进行聚类的方法，我们将其应用于数据验证问题。我们对分区进行摘要，并对这些摘要进行聚类。

机器学习的数据清洗。 众所周知，在构建机器学习模型的训练集时，数据清洗至关重要 [13, 20, 29, 61] 。大多数工具都需要人工输入，例如核验被预测为离群值的元组，这限制了工具的可扩展性 [14, 29, 61, 64]。Stonebraker 和 Ilyas [57], 也指出了可扩展性方面的顾虑，尤其是在企业中。Abedjan 等人 [1]讨论了在采用大规模数据集清洗方法时需要考虑的四类数据错误：离群值、重复项、规则违规（例如违反唯一性约束）以及模式违规（例如未通过类型检查）。我们的算法所使用的统计度量（例如完整性、频繁值、均值、标准差）涵盖了这些类别的数据错误。总体而言，数据清洗方法并不完全适用于我们的场景，因为我们并不打算清洗所有损坏的元组（我们的数据处理管道已经采用了性能较高的机器学习模型）。相反，我们只希望阻止模型基于损坏数据进行重新训练。此外，在我们所处理的规模下，难点在于识别出确实会导致模型性能下降的数据损坏，而不是识别任意数据损坏。

机器学习流水线的数据验证。 不同的数据错误会导致机器学习模型性能出现程度各异的下降，因此需要对机器学习流水线进行数据验证，以捕获严重错误 [8, 43, 51]。多个研究项目和开源软件库旨在验证机器学习流水线输入和输出中的数据。Biessmann 等人 [5] 讨论了数据验证的四个维度：正确性、一致性、完整性和统计属性；许多数据验证方法会随时间监控这些统计量。现有工作——例如 Deequ [52], TFX [4], DataSentinel [58], 和 DaQL [18]——定义了用于监控流水线输入和输出（例如特征和预测）的约束与库；然而，用户必须以声明式方式指定约束值（例如完整性界限），这对于我们这种拥有数千个数据集、且每个数据集包含数万个特征的场景并不实际。数据“ lint”工具通常会执行类型检查、重复项检测，并根据与均值相差固定个标准差的范围检测异常值，但它们不会直接关联下游机器学习模型性能，也无法预判模型性能下降 [18, 24]。在一项关于数据质量监控工具的广泛调查中，Ehrlinger 和 Wöß [19] 发现，大多数工具都需要数据的“黄金标准”，而这在包括我们所处环境在内的大多数生产环境中往往并不存在。此外，要将这些工具应用于持续场景，需要手动微调和监督；以我们的规模而言，这是无法做到的 [19]。Lwakatare 等人[33] 介绍了一项将数据验证工具集成到机器学习流水线中的案例研究，并发现实际集成过程十分棘手，而且大多数工程团队在工作流中都会忽略数据验证警报。例如，基于分布差值的双样本统计检验（例如 Wasserstein）经常触发大量误报 [34, 38, 59]。Lwakatare 等人 [33] 指出，这些警报没有提供可操作反馈（即没有指出哪些元组或列存在问题），而且需要过多人工维护（例如以声明式方式指定和微调约束）。Redyuk 等人 [48] 针对 PS 场景，形式化定义了“动态数据摄取”（即为不同应用摄取并验证新的数据分区）的自动化问题。他们提出了一种自动化的 k近邻方法，用于验证数据分区。我们的方法受到 Redyuk 等人 [48], 的启发，专门针对机器学习流水线场景，旨在预判下游机器学习性能下降。

数据清洗与验证方法评估。 以往关于数据清洗与验证的研究采用了多种评估度量指标 [9, 11, 27, 37, 42, 48, 51]。Koumarelas 等人 [27] 和 Pochampally 等人 [42] 选择精确率-召回率曲线下面积（auPRC）。我们使用平均精度（AP）而非 auPRC：auPRC 的估算通常使用梯形法则计算面积，这可能会过于乐观 [7]。另一些研究选择受试者工作特征曲线下面积（auROC）作为度量指标，或考察真正率（TPR）与假正率（FPR）之间的关系。尽管 auROC 度量指标能更好地体现方法的稳定性，但它无法恰当地反映精确率与召回率之间的权衡 [15, 31]。Davis 和 Goadrich [15] 讨论了类别分布发生偏斜时 ROC 曲线如何高估方法的性能，因为假阳性数量的大幅变化可能只会导致 ROC 曲线的 FPR 小幅变化。与非故障（即性能保持不变）相比，大多数机器学习流水线中的故障（即性能下降）数量相对较少；因此，故障的分布存在类别不平衡或偏斜。

8 结论

本文讨论了如何在下游机器学习模型性能下降之前自动验证数据。我们概述了生产机器学习流水线，以及工业规模下机器学习专用数据验证的需求。我们介绍了分区摘要（PS）数据验证方法，其中会对

带时间戳的分区进行比较，以确定异常分区。我们介绍了将现有数据验证方法一般性适配到 PS 场景的方法，以及 gate——一种无需工程师手动调优即可生成高精确率警报的方法。最后，我们讨论了在 Meta 的生产机器学习流水线中实施自动数据验证的经验。

参考文献

[1] Ziawasch Abedjan、Xu Chu、Dong Deng、Raul Castro Fernandez、Ihab F. Ilyas、Mourad Ouzzani、Paolo Papotti、Michael Stonebraker 和 Nan Tang。2016年。检测数据错误：现状如何，还有哪些工作需要完成？《VLDB基金会会刊》 9, 12 2016 993 1004 10 14778 2994509 2994518（8月），–。[2] https://doi.org/。/。Ziawasch Abedjan、Lukasz Golab 和 Felix Naumann。数据剖析：教程。载于2017年ACM数据管理国际会议论文集（美国伊利诺伊州芝加哥）（SIGMOD ’17）。计算机协会，纽约州纽约，美国, 2017 1747 1751 10 1145 3035918 3054772 –。[3] https://doi.org/。/。Fabrizio Angiulli 和 Clara Pizzuti。高维空间中的快速离群点检测。载于 PKDD。[4] Denis Baylor、Eric Breck、Heng-Tze Cheng、Noah Fiedel、Chuan Yu Foo、Zakaria Haque、Salem Haykal、Mustafa Ispir、Vihan Jain、Levent Koc 等。2017年。TFX：基于 TensorFlow 的生产规模机器学习平台。载于第23届 ACM SIGKDD知识发现与数据挖掘国际会议论文集。1387–1395。[5] Felix Biessmann、Jacek R. Golebiowski、Tammo Rukat、Dustin Lange 和 Philipp Schmidt。2021。机器学习系统中的自动化数据验证。《IEEE数据工程通报》44（2021），51–65。[6] Ane Blázquez-García、Angel Conde、Usue Mori 和 Jose A Lozano。2021。时间序列数据中的离群点/异常检测综述。《ACM计算调查》（CSUR）54，3（2021），1–33。[7] Kendrick Boyd、Kevin H Eng 和 C David Page。2013。精确率-召回率曲线下面积：点估计与置信区间。载于欧洲机器学习与数据库知识发现联合会议。施普林格，451–466。[8] Eric Breck、Marty Zinkevich、Neoklis Polyzotis、Steven Whang 和 Sudip Roy。2019。机器学习的数据验证。载于 SysML 论文集。https://mlsys.org/Conferences/2019/doc/2019/167.pdf [9] Laurens Castelijns。2019。PyWash：机器学习数据清洗助手。第7期（2019年7月）。[10] Varun Chandola、Arindam Banerjee 和 Vipin Kumar。2009年。异常检测：综述。《ACM计算调查》41，3，文章15（2009年7月），58页。https://doi.org/ 10 1145 1541880 1541882 。/。Zhaoqiang Chen、Qun Chen、Boyi Hou、Zhanhuai Li 和 Guoliang Li。2020年。迈向可解释、可学习的实体消歧风险分析。载于2020年ACM SIGMOD国际数据管理会议论文集（美国俄勒冈州波特兰）（SIGMOD ’20）。计算机协会，纽约州纽约，美国, 1165 1180 10 1145 3318464 3380572 –。[12] https://doi.org/。/。Heng-Tze Cheng、Levent Koc、Jeremiah Harmsen、Tal Shaked、Tushar Chandra、Hrishi Aradhye、Glen Anderson、Greg Corrado、Wei Chai、Mustafa Ispir 等。2016年。推荐系统的宽深度学习。载于首届推荐系统深度学习研讨会论文集。7–10。[13] Xu Chu、Ihab F Ilyas、Sanjay Krishnan 和 Jiannan Wang。2016年。数据清洗：概述与新兴挑战。载于2016年数据管理国际会议论文集。2201–2206。[14] Xu Chu、John Morcos、Ihab F. Ilyas、Mourad Ouzzani、Paolo Papotti、Nan Tang 和 Yin Ye。2015年。KATARA：由知识库和众包驱动的数据清洗系统。载于2015年ACM SIGMOD国际数据管理会议论文集（澳大利亚维多利亚州墨尔本）（SIGMOD ’15）。计算机协会，纽约州纽约，美国，1247–1261。https://doi.org/ 10 1145 2723372 2749431 。/。Jesse Davis 和 Mark Goadrich。精确率-召回率曲线与 ROC 曲线之间的关系。载于第23届国际机器学习会议论文集（美国宾夕法尼亚州匹兹堡）（ICML ’06）。计算机协会，纽约州纽约，美国，233–240。https://doi.org/10.1145/1143844.1143874[16] Denis Moreira dos Reis、Peter Flach、Stan Matwin 和 Gustavo Batista。2016年。使用增量式柯尔莫哥洛夫-斯米尔诺夫检验进行快速无监督在线漂移检测。载于第22届 ACM SIGKDD知识发现与数据挖掘国际会议论文集。1545–1554。[17] Connor Dowd。2020年。一种新的经验累积分布函数双样本检验统计量。arXiv：方法论（2020）。[18] Lisa Ehrlinger、Verena Haunschmid、Davide Palazzini 和 Christian Lettner。2019年。一种用于监控机器学习应用程序数据质量的 DaQL。载于 DEXA。[19] Lisa Ehrlinger 和 Wolfram Wöß。2022。一项关于数据质量度量和监控工具的调查。《大数据前沿》（2022），28。[20] Venkat Gudivada、Amy Apon 和 Junhua Ding。2017年。大数据和机器学习的数据质量考量：超越数据清洗与 2006 [15] 其他。[21] Joseph M Hellerstein. 2008. 大型数据库的定量数据清洗。In Ellen – Hohma、Christian M. M. Frey、Anna Beer 和 Thomas Seidl. 2022. SCAR：加速并增强稳健性的谱聚类。《VLDB基金会会刊》15, 11 (2022年9月), 3031–3044. https://doi.org/10.14778/3551793.3551850[23] Ling Huang、XuanLong Nguyen、Minos Garofalakis、Michael Jordan、Anthony Joseph 和 Nina Taft. 2006年. 网络内主成分分析与异常检测。神经信息处理系统进展19 (2006). [24] Nick Hynes、D. Sculley 和 Michael Terry. 2017年. 数据检查器：面向机器学习数据集的轻量级自动合理性检查。http://learningsys.org/nips17/ assets/papers/paper_19.pdf [25] Firuz Kamalov 和 Ho Hon Leung. 2020年. 高维数据中的离群点检测。《信息与知识管理期刊》19, 01 (2020年3月), 2040013. https://doi.org/ 10 1142 0219649220400134 . /s Daniel Kang、Deepti Raghavan、[26] Peter Bailis 和 Matei Zaharia. 2020年. 用于监控和改进机器学习模型的模型断言。载于《机器学习与系统论文集》，I. Dhillon、D. Papailiopoulos 和 V. Sze（编），第2卷。481–496. https://proceedings.mlsys.org/paper/2020/file/ a2557a7b2e94197ff767970b67041697-Paper.pdf [27] Ioannis Koumarelas、Lan Jiang 和 Felix Naumann. 2020年. 用于重复项检测的数据准备。《数据与信息质量期刊》12, 3, 文章15 (2020年6月), 24 10 1145 3377878 页。[28] https://doi.org/ . / Sanjay Krishnan、Michael J Franklin、Ken Goldberg 和 Eugene Wu. 2017年. Boost-clean：面向机器学习的自动错误检测与修复。arXiv预印本 arXiv:1711.01299 (2017). [29] Sanjay Krishnan、Jiannan Wang、Eugene Wu、Michael J Franklin 和 Ken Goldberg. 2016年. Activeclean：用于统计建模的交互式数据清洗。《VLDB基金会会刊》9, 12 (2016), 948–959. [30] Kaiyu Li 和 Guoliang Li. 2018. 近似查询处理：有哪些新进展，未来方向何在？《数据科学与工程》3 (2018), 379–397. [31] Jorge M Lobo、Alberto Jiménez-Valverde 和 Raimundo Real. 2008. AUC：预测分布模型性能的一种误导性度量。《全球生态学与生物地理学》17, 2 (2008), 145–151. [32] Ning Lu、Guangquan Zhang 和 Jie Lu. 2014. 通过能力模型检测概念漂移。《人工智能》209 (2014), 11–28. [33] Lucy Ellen Lwakatare、Ellinor Rånge、Ivica Crnkovic 和 Jan Bosch. 2021. 工业机器学习项目中采用自动化数据验证的经验。载于《2021年IEEE/ACM第43届国际软件工程会议：软件工程实践》（ICSE-SEIP）。IEEE, 248–257. [34] Frank J Massey Jr. 1951. 拟合优度的柯尔莫哥洛夫–斯米尔诺夫检验。《美国统计学会会刊》46, 253 (1951), 68–78. [35] Saeed Mehrang、Elina Helander、Misha Pavel、Angela Chieh 和 Ilkka Korhonen. 2015. 联网体重秤体重时间序列中的离群点检测。载于《2015年IEEE国际生物信息学与生物医学会议》（BIBM）。IEEE, 1489–1496.[36] Guido Moerkotte. 1998. 小型物化聚合：一种轻量级数据仓储索引结构。载于《超大型数据库会议》。[37] Barzan Mozafari、Purna Sarkar、Michael Franklin、Michael Jordan 和 Samuel Madden. 2014. 将众包扩展至超大型数据集：主动学习的用武之地。《VLDB基金会会刊》8, 2 (2014年10月), 125–136. https://doi.org/10. 14778/2735471.2735474[38] Alfred Müller. 1997. 积分概率度量及其生成函数类。《应用概率进展》29, 2 (1997), 429–443. http://www. jstor.org/stable/ 1428011 [39] 1983 Paul Newbold. . ARIMA模型构建与时间序列分析预测法。《预测期刊》2, 1 (1983), 23–35. [40] Shimei Pan 和 Tao Ding. 2019. 基于社交媒体的用户嵌入：文献综述。arXiv预印本 arXiv:1907.00725 (2019). [41] Matthew Partridge 和 Rafael A Calvo. 1998. 快速降维与简易主成分分析。《智能数据分析》2, 3 (1998), 203–214.[42] Ravali Pochampally、Anish Das Sarma、Xin Luna Dong、Alexandra Meliou 和 Divesh Srivastava. 2014. 利用相关性融合数据。载于《2014年ACM SIGMOD国际数据管理会议论文集》（美国犹他州斯诺伯德）（SIGMOD ’14）。计算机协会，纽约州纽约, 433 444 10 1145 2588555 2593674 –。https://doi.org/ . / . [43] Neoklis Polyzotis、Sudip Roy、Steven Euijong Whang 和 Martin Zinkevich. 2018. 生产环境机器学习中的数据生命周期挑战：综述。《SIGMOD记录》47, 2 2018 17 28 10 1145 3299887 3299891 (2018年12月), –。[44] https://doi.org/ . / . Stephan Rabanser、Stephan Günnemann 和 Zachary Lipton. . 高声警：检测数据集偏移方法的实证研究。《神经信息处理系统进展》32 (2019). [45] Erhard Rahm 和 Philip Bernstein. 2001. 自动模式匹配方法综述。《VLDB期刊》10 (2001年12月), 334–350. https://doi.org/10.1007/ s007780100057[46] Sridhar Ramaswamy、Rajeev Rastogi 和 Kyuseok Shim. 2000. 从大型数据集中挖掘离群值的高效算法。《SIGMOD记录》29, 2 (2000年5月), 427–438. https://doi.org/10.1145/335191.335437

[47] Zhu M Recall. 2004. 精确率与平均精度。统计与精算科学系。滑铁卢大学，滑铁卢 (2004). [48] Sergey Redyuk、Zoi Kaoudi、Volker Markl 和 Sebastian Schelter. 2021. 面向动态数据摄取的自动化数据质量验证。载于*EDBT*。[49] Marco Tulio Ribeiro、Sameer Singh 和 Carlos Guestrin. 2016年. 机器学习的模型无关可解释性。*arXiv*预印本 *arXiv:1606.05386* (2016年). [50] Peter J Rousseeuw 和 Mia Hubert. 2011. 用于离群点检测的稳健统计。*Wiley*跨学科评论：数据挖掘与知识发现 1, 1 (2011), 73–79.[51] Sebastian Schelter、Dustin Lange、Philipp Schmidt、Meltem Celikel、Felix Biessmann 和 Andreas Grafberger. 2018. 大规模数据质量验证自动化。《*VLDB*基金会会刊》11, 12 (2018年8月), 1781–1794. https: // doi.org/10.14778/3229863.3229867[52] Sebastian Schelter、Philipp Schmidt、Tammo Rukat、Mario Kiessling、Andrey Taptunov、Felix Biessmann 和 Dustin Lange. 2018. DEEQU——面向机器学习流水线的数据质量验证。[53] Raghav Sethi、Martin Traverso、Dain Sundstrom、David Phillips、Wenlei Xie、Yutian Sun、Nezih Yegitbasi、Haozhun Jin、Eric Hwang、Nileema Shingte 等. 2019. Presto：万物皆可运行SQL。载于*2019*年*IEEE*第*35*届国际数据工程大会（*ICDE*）。IEEE, 1802–1813. [54] Shreya Shankar、Rolando Garcia、Joseph M. Hellerstein 和 Aditya G. Parameswaran. 2022年. 机器学习运营化：一项访谈研究。https://doi.org/ 10 48550 2209 09125 . /ARXIV. . Shreya [55] Shankar 和 Aditya Parameswaran. 2021. 实现机器学习流水线的可观测性。*arXiv*预印本 *arXiv:2108.13557* (2021年). [56] Michael Stonebraker、Daniel Bruckner、Ihab F. Ilyas、George Beskales、Mitch Cherniack、Stanley B. Zdonik、Alexander Pagan 和 Sha Xu. 2013. 大规模数据整理：Data Tamer系统。载于*CIDR*。[57] Michael Stonebraker 和 Ihab F. Ilyas. 2018. 数据集成：现状与未来方向。《*IEEE*数据工程通报》41 (2018), 3–9.

[58] Arun Swami、Sriram Vasudevan 和 Joojay Huyn。2020年。Data Sentinel：生产级数据验证平台。在*2020*年 *IEEE* 第*36*届国际数据工程会议（*ICDE*）上。1579–1590。https://doi.org/10.1109/ ICDE48307.2020.00140[59] SS Vallender。1974年。计算直线上概率分布之间的 Wasserstein距离。《概率论及其应用》18, 4 (1974), 784–786。[60] Scott White 和 Padhraic Smyth。2005年。一种在图中发现社群的谱聚类方法。在*2005*年 *SIAM* 国际数据挖掘会议论文集中。SIAM，274–285。[61] Weiyuan Wu、Lampros Flokas、Eugene Wu 和 Jiannan Wang。2020年。面向 query 2.0 的投诉驱动训练数据调试。在*2020*年 *ACM SIGMOD*国际数据管理会议论文集中。1317–1334。[62] Doris Xin、Hui Miao、Aditya Parameswaran 和 Neoklis Polyzotis。2021年。生产环境中的机器学习数据处理管道：实证分析与优化机遇。在*2021*年国际数据管理会议论文集（线上活动，中国）（*SIGMOD '21*）中。计算机协会，纽约，纽约州，美国，2639 2652 10 1145 3448016 3457566 –。[63] 2005 https://doi.org/。/。Rui Xu 和 Donald Wunsch。聚类算法综述。*IEEE* 神经网络汇刊16, 3 (2005), 645–678。[64] Mohamed Yakout、Ahmed K. Elmagarmid、Jennifer Neville、Mourad Ouzzani 和 Ihab F. Ilyas。2011。引导式数据修复。《*VLDB*基金会会刊》4, 5（2011年2月），279 289 10 14778 1952376 1952378 –。https://doi.org/。/。[65] Tian Zhang、Raghu Ramakrishnan 和 Miron Livny。1996。BIRCH：一种面向超大型数据库的高效数据聚类方法。在*1996*年 *ACM SIGMOD*国际数据管理会议论文集（加拿大魁北克蒙特利尔）（*SIGMOD '96*）中。计算机协会，纽约，纽约州，美国，103–114。https://doi.org/10.1145/233269.233324