摘要

如今，机器学习（ML）在我们日常生活的许多方面都发挥着至关重要的作用。 从本质上讲，构建性能良好的 ML 应用程序需要在此类应用程序的整个生命周期中提供高质量的数据。 然而，现实世界中的大多数表格数据都存在不同类型的差异，如缺失值、离群值、重复数据、模式偏差和不一致性。 这些差异通常是在收集、传输、存储和/或整合数据时出现的。 为了处理这些差异，人们引入了许多数据清理方法。 然而，这些方法大多忽略了下游 ML 模型的要求。 因此，在 ML 管道中利用这些数据清理方法的潜力尚未得到充分挖掘。 在这项工作中，我们引入了一个名为 REIN1 的综合基准，以彻底研究数据清洗方法对各种 ML 模型的影响。 通过该基准，我们为一些重要的研究问题提供了答案，例如数据清理在哪些方面以及是否是 ML 管道的必要步骤。 为此，该基准测试了 38 种简单和先进的错误检测与修复方法。 为了对这些方法进行评估，我们使用了大量在 14 个公开数据集上训练过的 ML 模型，这些数据集涵盖不同领域，既有真实的错误剖面，也有合成的错误剖面。

1 引言

随着现代计算技术的发展，当今许多行业都在开发强大的 ML 模型，这些模型能够分析庞大而复杂的数据，同时在巨大的规模上提供快速而准确的结果。 组织和企业通常会利用这些结果做出更好的决策，而无需或只需极少的人工干预。 然而，此类决策的正确性在很大程度上取决于可用数据的质量。 根据 Gartner 最近的一项研究[37]，企业认为数据质量不佳平均每年造成 1 500 万美元的损失。 IBM 2016 年的另一项研究[45]显示，数据质量差每年给美国经济造成 3.1 万亿美元的损失。 这些研究表明，数据质量问题主要是代价高昂且普遍存在。

几十年来，数据质量一直是一个活跃的研究领域。 在这种情况下，数据管理社区将数据质量问题作为 ETL 工作流程的一部分来解决。 因此，人们提出了许多自动检测和/或修复数据差异的建议[10, 12, 20, 32, 44, 46]。 事实上，这些建议中只有一小部分在发现和修复错误实例时考虑了数据错误的异质性特征。 换句话说，大多数建议的技术只针对一种错误类型。 此外，大多数此类方法都是在脱离下游 ML 应用的情况下开发的。 因此，在预测任务中采用此类清理方法的后果被广泛掩盖。 因此，在 ML 管道中选择最合适的清理策略（即错误检测和修复方法的组合）成为一项挑战。

在本文中，我们通过引入一个称为 REIN 的基准框架来应对这一挑战。 REIN 的主要目标是深入研究数据清理与 ML 建模之间的相互作用。 通过大量实验，REIN 研究了大量与各种 ML 模型相结合的清理策略，包括分类、回归、聚类和 AutoML 模型。 在 REIN 中，我们对错误检测和修复方法进行了评估，这些方法既可以作为独立方法使用，也可以作为 ML 管道中的组件使用。 为此，有必要掌握现有脏数据集的基本事实。 然而，要找到同

样适用于人工智能任务的此类数据集通常并不容易。 开展此类综合研究的另一个挑战是所需实验的规模。 由于涉及大量错误检测和修复方法，需要训练的模型数量激增（参见第 2 节）。 对于这些检测和修复方法来说，提供必要的配置和信号（即模式、规则和帮助函数）也至关重要。 最后，ML 模型本质上是概率模型，重新采样可能会改变结果。 因此，我们需要验证从 ML 实验中获得的结论。

具体而言，本文的贡献如下：(1) 我们定义了一个架构框架，用于系统地评估专用于表格数据的错误检测和修复工具。 除了相对于地面实况的传统评估方法外，REIN 还能让数据科学家和从业人员利用多个预测模型的性能来正确判断其检测和修复方法。 此外，REIN 还利用交集大于联合（IoU）指标来量化数据清理方法之间的相似性。 (2) 我们设计了一个基准控制器，可有效管理框架中的其他组件。 这种控制器利用设计时的知识优势，例如错误类型和 ML 任务，广泛地避开了不必要的实验，从而降低了运行基准的复杂性。 (3) 我们根据方法论和所需配置对最著名的错误检测和修复方法进行了分类。 (4) 我们根据数据版本（即地面实况、脏数据或修复数据）的特点，检查了相关 ML 模型在不同场景下的性能。 (5) 我们通过使用小型、中型和大型数据集作为这些方法的输入，评估误差数据清理方法的可扩展性。 此外，我们还通过重复实验来评估这些方法的鲁棒性，同时单调地增加误差率。 (6) 我们采用带有连续性校正的 Wilcoxon 符号秩检验来补偿训练过程中的随机性。 据我们所知，REIN 是第一个从检测和修复性能、预测准确性、鲁棒性和可扩展性等不同角度评估数据清理方法的大规模基准框架。

2 基准概述

在本节中，我们将介绍 REIN 的架构以及我们的假设。REIN 包含多个数据处理和评估步骤。 具体来说，几个脏数据集 Φ- =ϕ1-,---,ϕn-,ϕi- ∈Ru×v 被用作不同顺序向量的输入。

向量α1, - - - , αm，其中上标'-'表示脏数据集，ϕu×v表示脏数据集。u, v分别表示u, v中的记录数和属性数。之后，由每种检测方法识别出的错误实例会用修复方法替换。

这一步骤的结果是+ =ϕ+ ,--- ,ϕ+，其中ϵ =m×k i,1 i,ϵ 表示每个脏数据集 k 生成的修复版本的数量。上标 "+"表示已修复数据集。 最后，对每个修复后的数据集 ϕ+ 进行采样，以训练多个 ML i,j 模型 ϕ+ 。模型 γ1, - - - , γh ，其中 h 是所涉及的 ML 模型数量。 因此，每个脏数据集 ϕi- 的 ML 实验次数为 (ϵ + 1) × h × s，其中每个实验重复 s 次，以估计均值和标准偏差，并将脏版本添加到生成的修复版本中。

为了实现这样的大规模基准，我们采用了图 1 所示的架构。

架构如图 1 所示。 数据存储库，即 Post-

greSQL 数据库，用于存储地面实况 Φ、g

脏数据 Φ- 和生成的修复版本集 Φ+。 为了正确控制实验，错误注入模块会生成不同类型、不同错误率的错误。 实际上，错误注入任务是在运行实验前的离线阶段进行的（详情参见第 5 节）。 另一个组件是数据清理工具箱，它是一个包含所有可用错误检测和重新配对工具的池。 其中一些工具，如 NADEEF、HoloClean 和 OpenRefine，如果不提供一组清洗信号，就无法使用。 这些信号包括功能依赖性约束、完整性约束、知识库、模式和预估配置。

REIN 的主要组件是基准控制器、

它连接基准中的所有其他组件。 这种控制器的

这种控制器有三个目的： 首先，它能在不同模块之间平滑交换地面实况 Φ、脏数据 Φ- 和修复数据 g Φ+ 。

Φ+ 在不同模块之间顺利交换。 其次，它利用有关脏数据集的先验知识，避免了不必要的错误检测和修复操作。 例如，如果已知某个数据集有重复数据（如引文数据集），那么运行违反规则或异常值检测方法就毫无意义。 第三，它利用先验知识来调整数据准备步骤，以适应相关的 ML 任务。 架构的最后一个组件是评估模块，它为错误检测和修复方法以及 ML 模型提供服务。 评估模块利用多个质量指标来估算在地面实况、脏数据和修复数据上训练的 ML 模型的预测性能。

另一个组件是由分类、回归和聚类方法组成的 ML 模型库。 此外，REIN 还检查了两种 AutoML 算法，以检查由数据清理和建模模块组成的全自动管道的性能。 最后，一个评估模块从准确性、延迟、可扩展性和鲁棒性等四个指标来检查数据清理和建模方法的性能（参见第 6 节）。 由于篇幅有限，我们在源代码的 README 文件中定义了以下内容： (1) 如何在有/没有脏数据集基本事实的情况下运行基准；(2) 如何通过添加新数据集、ML 模型和数据清理工具来轻松扩展 REIN 框架。

3 数据清理方法

在本节中，我们将概述所研究的错误检测和修复方法。

3.1 错误检测方法

在 REIN 中，我们选择了 19 种公开的错误检测方法，这些方法可处理表格数据中最常见的属性和类错误 2 。 表 1 列出了错误检测方法及其针对的错误类型。 此外，该表还包括运行每种错误检测方法所需的配置和/或信号，即模式、约束、帮助函数和知识库。 在 REIN 中，我们根据错误检测方法的方法论将其分为两大类，包括（I）非学习方法和（II）ML 支持方法。 顾名思义，前一类包括使用一套用户提供的知识库、业务规则、完整性约束或使用一套统计措施来检测错误的方法和工具。 这些方法和工具通常都能解决特定的错误类型，如重复、异常值或缺失值。 第二类包括 Picket、ED2 和 RAHA 等方

法，它们将错误检测任务表述为分类问题。 这些方法最初为每个属性提取一组特征。 这些自动生成的特征能让分类器区分干净和不干净的数据样本。 为了训练这样的分类器，需要选择一些训练样本，由 Oracle 进行标记。 下面，我们将介绍各类错误检测器。

非学习检测器： 表 1 中的第一种方法是 KATARA [10]，它将输入的脏数据集与众包知识库进行比对，以识别并纠正与语义模式不符的数据样本。 为了检测违反规则和模式的情况，NADEEF[12]通过提供实施拒绝约束和其他用户定义函数的接口，全面处理数据质量规则。 另一项相关工作是 HoloClean [46]，它将定性和定量信号（如拒绝约束和相关性）结合到一个统计模型中，从而能够检测和修复缺失值以及规则/约束违规。 为了识别不一致和违反模式的情况，OpenRefine 工具[19] 使用户能够通过分面和过滤操作直观地探索脏数据集。 FAHES [44] 是另一种检测伪装缺失值的工具，例如电话号码中的 "99999"。 为此，FAHES 对分类数据采用了语法模式检测模块，对数值数据采用了基于密度的离群值检测模块。 为了检测明确的缺失值，REIN 采用了一种方法来查找空条目或 NAN 条目。

dBoost [34] 是一种离群点检测方法，它整合了几种最广泛应用的离群点检测算法，包括直方图、高斯和多元高斯混合物。为找到此类算法的最佳超参数，dBoost 采用随机搜索，搜索空间为所有可能的配置。 其他离群点检测方法包括标准差 (SD)、四分位距 (IQR) [55] 和隔离森林 (IF) [30]。 前一种方法将单元格 $x \in A$（其中 A 表示属性）标注为离群值，如果该单元格与 A 中条目平均值相差 n 个标准差数。 一个更具抵抗力的统计量是 IQR，定义为属性 A 的第 25 个百分位数和第 75 个百分位数之差，即 $IQRA = Q3 - Q1$。 在这种情况下，离群值是指超出$[Q1 - k \times IQRA, Q3 + k \times IQRA]$ 范围的任何值，其中k 和n 是超参数。后一种方法的目标是在不分析所有数据样本的情况下识别异常值。具体来说，IF 方法会为脏数据集构建一个隔离二叉树集合，而异常值则是二叉树上平均路径长度较短的数据样本。

为了检测重复数据，REIN 研究了两种方法，即 Key Collision [29] 和 ZeroER [54]。 前一种方法需要用户提供假定唯一的关键属性信息。 在这种情况下，只要两条记录的关键属性值相同，就能检测出它们是重复的。 后一种方法依赖于 Magellan [24] 来生成一组相似性特征。 不过，ZeroER 需要零标记的示例，它通过高斯混合模型（Gaussian Mixture Model）来学习匹配和不匹配特征向量的分布。 除了重复数据，CleanLab[39] 还利用自信学习原理来估计噪声标签和真实标签的联合分布，从而检测噪声标签。 为了解决数据错误的异质性问题，Min-K 和 Max Entropy [2] 采用了其他非学习方法的组合来识别数据集中现有的大部分错误样本。 具体来说，Min-K 将至少有 k 种方法检测到的样本视为错误。

另外，最大熵（Max Entropy）引入了一种基于熵的取样方法，用于系统地选择非学习方法的执行顺序。

ML 支持的检测器： REIN 中研究的 ML 支持方法在生成特征的方式和减少所需标记预算的方式上有所不同。 例如，元数据驱动的错误检测方法[53]实现了一个元数据生成器和一套非学习错误检测方法来提取特征。 在这种情况下，每种非学习方法都由一个二元特征表示，如果非学习方法识别出该单元格是脏的，则特征值为 1。 为了减少标签预算，RAHA [33] 采用了一种半监督算法，该算法通过相似性对样本进行聚类，并按聚类获取标签，然后再在每个聚类中传播获取的标签。 同样，ED2 [38] 提取了一组属性级、元组级和数据集级特征，这些特征定义了数据集的分布。 此外，ED2 还利用主动学习来获取分类器不确定的干净/错误样本标签。 最后，Picket [31] 采用自监督方法训练错误检测模型，无需用户标签。

3.2 数据修复方法

在 REIN 中，我们研究了 19 种数据修复方法，这些方法可根据其干预类型分为两大类，即 (I) 通用方法和 (II) 面向 ML 的方法。 前一类包括直接修改脏数据集以生成修复版本的方法。 这种修改可以是删除脏单元，也可以是用一组生成的修复单元来替代脏单元。 它们具有通用性，即无论下游应用（如 ML 建模、数据可视化或数据富集）如何，它们都力求提高数据质量。 另外，第二类方法包括联合优化数据质量和下游 ML 模型性能的方法。 在 REIN 中，我们还利用脏数据的地面实况来显示性能上限。 下面，我们将介绍每个类别中的各种方法。

通用修复方法。 为了生成修复值，REIN 研究了几种标准和 ML 驱动的估算方法。 标准估算方法利用简单的统计量（如平均值、中位数或模式）来生成数值的修复值。 对于分类值，我们只需利用模式，即相应属性中最常出现的值，作为修复值。 高级估算方法是建立 ML 模型，根据整个数据集的信息生成准确修复值的方法。 对于数值，REIN 研究了 5 种基于 ML 的估算方法，包括 K 近邻（KNN）、决策树（DT）、贝叶斯岭（Bayesian Ridge）[42]、基于随机森林（RF）的 missForest [51] 和基于深度神经网络的 DataWig [7]。 对于分类值，我们同时研究了 miss Forest 和 DataWig。 其中，missForest 在预测缺失值之前，首先在一组干净样本（即没有异常值的完整样本）上迭代训练 RF 模型。 同样，DataWig 将深度学习模块与神经架构搜索和端到端归因管道优化相结合。

对于混合类型数据集，missForest 和 DataWig 有两种运行模式，即单独模式和混合模式。在前一种模式下，每种方法针对每种数据类型单独执行，称为 MISS-Sep；而在后一种模式下，考虑到不同数据类型之间可能存在的关系，每种方法针对所有数据类型整体执行，称为 MISS-Mix 和 DataWig-Mix。 另一种通用方法是 HoloClean [46]，它通过整体采用多个清洗信号来建立概率图模型，从而精确推断出修复值。 为了修复模式违规和不一致，OpenRefine[19]利用谷歌提炼表达式语言（GREL）作为其母语来转换现有数据或创建修复值。 这一类中的最后一种方法是 BARAN [32]，它是一种修复所有错误类型的整体无配置方法。 为此，BARAN 会训练可增量更新的模型，利用数据错误的值、邻近地区和领

域上下文来提出修正候选。 为了进一步增加训练数据，BARAN 利用了维基百科页面修订历史等外部资源。

面向 ML 的修复方法： 第二类包括旨在联合优化清理和修改任务的方法。 换句话说，这些方法侧重于选择最佳修复候选对象，目的是提高特定预测模型的性能。 因此，这些方法假定可以从其他通用方法中获得修复候选数据。 例如，BoostClean [25] 将纠错任务视为一个统计提升问题，即一组弱学习者组成一个强学习者。 为了生成弱学习器，BoostClean 会迭代选择一对检测和修复方法，然后将它们应用于训练集，从而生成一个新模型。 ActiveClean [26] 是另一种面向 ML 的方法，主要用于具有凸损失函数的模型。 它将数据清理任务表述为一个随机梯度下降问题。 起初，它在一个脏的训练集上训练一个模型，并对该模型进行迭代更新，直到达到全局最小值。 在每一次迭代中，ActiveClean 都会对一组记录进行采样，然后要求神谕对其进行清理，以沿着最陡峭的梯度移动模型。 类似的工作还有 CPClean [22]，它以增量的方式清理训练集，直到确定没有更多的修复可能改变模型预测为止。

4 数据建模

在本节中，我们将介绍一组具有代表性的常用 ML 模型，用于评估错误检测和修复方法的性能。 表 2 总结了这些算法，以及它们是用于分类 (C)、回归 (R) 还是无监督聚类 (UC) 任务。 如表中所列，REIN 研究了 12 个分类器、11 个回归模型、6 个聚类算法和 2 个 AutoML 算法。 这些重要算法广泛适用于现实世界的各种应用领域，如网络安全系统、智能城市、医疗保健、电子商务、农业文化等[49]。 使用这两种 AutoML 算法的理由是为了评估由数据清理和模型构建组成的完全自动匹配的 ML 管道的性能。 我们感兴趣的是检查这些算法是否能够在脏数据集或自动修复数据集的基础上找到模型架构和超参数的最佳组合。 对于大多数模型，REIN 利用 Scikit-learn [42] 库的 Python 版本进行训练和测试。 在超参数调整方面，REIN 利用了一种基于贝叶斯的知情搜索方法，即 Optuna [3]。 不过，我们没有将 Optuna 与 AutoML 算法一起使用，因为它们可以自动选择最佳超参数。 此外，我们也没有使用这些算法的内部处理流水线，因为我们主要关注的是已检查过的清理器（如表 1 所列）。

在 REIN 中，我们评估了五种情况下的各种错误检测和修复方法。 表 3 总结了根据用于训练和测试的数据版本定义的不同场景。 除了脏数据和修复后的数据版本外，我们还利用地面实况版本来估算性能上限。 例如，S1 涉及在脏版或修复版数据上训练和测试 ML 模型。 反之，S4 代表最佳设置，即使用地面实况版本的数据来训练和测试模型。 为了捕捉仅在一个阶段就能实现最佳数据清理的性能，REIN 还考虑了 S3 和 S4，在这两个阶段

中，地面真实数据（模拟最佳数据清理）分别用于训练和测试。 最后，S5 主要用于以 ML 为导向的修复方法，其输出是生成 ML 模型。

一般来说，由于 ML 的随机性，在每个方案中获得的结果可能会有所不同。 因此，在得出结论之前，仔细研究每种情况下获得的结果至关重要。 在这方面，REIN 利用 A/B 假设检验来提高我们对所得结果解释的信心。 一般来说，A/B 假设检验可用于量化在样本分布相同的假设条件下观察到两个数据样本的可能性[14]。 在 REIN 中，A/B 假设检验可以从统计学角度预测 ML 模型在不同情况下的表现是否相似。 测试程序的第一步是明确界定零假设$H0$ 和原生假设Ha。 在 REIN 中，零假设$H0$ 是指一个 ML 模型在两个不同的场景（如 S1 和 S4）中具有大致相同的性能，与数据版本无关。 反之，备择假设Ha 则表示 ML 模型在 S1 和 S4 中表现不同。 统计意义用 p 值来估算，即 S1 和 S4 之间观察到的差异可能是随机发生的概率。 为了估算 p 值，我们使用了非参数 Wilcoxon 符号秩检验[14]。 这种检验的主要优点在于不对抽样分布（如高斯分布）做任何假设。 具体来说，我们选择了双尾检验，因为事先并不知道 S1 和 S4 的结果差异是有利于 S1 还是 S4。 计算 p 值后，我们可以将其与显著性水平 α 进行比较，以估计是否拒绝零假设 $H0$。 具体来说，如果 p 值小于 α，我们可以拒绝零假设 $H0$。 否则，我们的结论是，在比较方案中得到的结果支持备择假设H。

5 基准数据

在本节中，我们将详细介绍真实世界的数据集以及如何在其中注入误差。 为了系统地选择运行基准的适当数据集，有必要根据 REIN 的目标确定一系列要求。 这些目标围绕着单独估算每种检测器/修复方法的性能，而不考虑 ML 管道的后续阶段，并检查这些方法在不同情况下对下游预测模型性能的影响。 因此，REIN 所涉及的数据集必须满足以下条件：(1) 存在完整、干净的地面实况版本；(2) 存在相关的预测任务，如分类、回归或聚类；(3) 存在不同的数据类型，如分类、数值和/或文本；(4) 存在不同的现实误差曲线。 事实上，我们收集的两个数据集，即 Beers 和 Citation，都满足这些条件。 然而，要找到满足我们要求的其他数据集并非易事。

为了克服这一难题，我们选择向一组真实世界的数据集中注入不同类型的错误。 因此，我们可以通过获取每个数据集的多个版本以及地面实况，对实验进行主要控制。 除了上述要求，我们还希望选择涵盖多个应用领域的数据集，例如商业、医疗和工业领域，因为不同领域的数据通常具有不同的特征。 此外，我们还选择了不同规模的数据集，从几百个样本到几十万个样本不等，以精确测试各种数据清理方法的效率。 表 4 总结了所考察的数据集及其地面实况的特征。

为了向真实世界数据集注入错误，REIN 利用了 BART 工具[5]，该工具可对错误数量和这些错误的修复难度进行系统控制。 为了使用 BART 注入错误，我们使用了一组拒绝约束

来生成不同的属性和类错误，如违反规则、异常值、空、重复和错误标签。 此外，我们还使用 Python 库（称为错误生成器）来生成高度逼真的错误 [1]。 这类错误的例子包括键盘上的错别字、隐含缺失值、高斯噪声和数值交换。 为了自动生成 FD 规则，REIN 借助了 FDX 剖析器[56]，该剖析器将学习功能依赖关系的任务表述为稀疏回归问题。 生成 FD 规则后，我们手动将其转换为否认约束，以便与 BART 和基于规则的错误检测和修复方法（如 HoloClean 和 NADEEF）一起使用。

6 性能评估

在本节中，我们将评估各种错误检测和修复方法的有效性和效率。 我们首先介绍评估的设置，然后讨论整个研究的结果和经验教训。

6.1 实验设置

在 REIN 中，我们利用多个指标来评估典型 ML 流水线不同阶段的结果质量。 在错误检测阶段，我们利用精确度、召回率、F1 分数、IoU 和运行时间来评估有效性和效率。 在这种情况下，精确度 P 表示检测到的实例中相关实例（如实际的错误单元格）的比例，即 $P = tp+fp$ 其中，tp 和 fp 分别为真阳性和假阳性。 召回率 R 定义为检测到的错误实例的百分比，即 $R =$

其中，fn 表示假阴性。 F1 分数表示精确度和召回率的调和平均值，其中

$F\,1 = 2.\,P\,.R$ 。 这些指标定义了相对于地面实况$P+R$ 的检测质量。 不过，识别不同检测方法检测到的错误单元之间的相似性也很重要。 因此，我们采用了 "交集大于联合"（IoU）度量。 假设Na、Nb 是检测器 a 和b 检测到的错误小区。 因此，探测器 a 和 b 之间的 IoU 指标计算公式为 $|Na \cap Nb|$ 。 $|Na|+|Nb|-|Na\cap Nb|$

对于这些计算，我们只考虑真阳性，因为假阳性可能会导致误导性结果。 最后，运行时间是指遍历整个数据集以识别错误单元格所花费的时间。

在错误修复阶段，我们要区分自然属性和分类属性。 对于前者，我们采用均方根误差（RMSE）作为修复值与基本真实值之间的距离值。 事实上，某些错误类型（如错别字和异常值）会将数字实例转化为分类实例。 为了正确计算 RMSE 指标，我们过滤掉了未被检测和修复的转化实例。 对于后一种数据类型，我们采用了精度、召回率和 F1 度量。 在这种情况下，精确度被定义为正确修复的数据错误占已修复数据错误数量的比例。 召回率定义为正确修复的数据错误相对于数据错误数量的百分比。 我们还报告了运行时间，以量化生成修复结果所耗费的时间。 在 ML 建模阶段，我们使用精度、召回率和 F1 指标来衡量分类模型。 对于需要将聚类数目k 作为输入的聚类方法，我们利用 Silhouette 指数来找到对k 值的良好估计。 在 A/B 统计检验中，我们将 I 类错误率α 设为 0.05。 所有实验都使用不同的随机种子重复了十次，这些种子控制着训练-测试的分割，我们报告

了十次运行的平均值。 我们在 Ubuntu 16.04 LTS 机器上运行所有实验，该机器有 32 个 2.60 GHz 内核和 264 GB 内存。 由于篇幅有限，许多实验结果被省略。

6.2 误差检测

在这组实验中，我们评估了应用于不同数据集的几种错误检测器的性能。 对于每个数据集，检测器的数量取决于注入错误的类型。 此外，图中还特意排除了未能检测到任何单元格的检测器。 图 2a 显示了 Beers 数据集中使用 14 种错误检测方法检测到的错误单元数（蓝柱）和真阳性单元数（绿柱）。 假阳性的数量通过将蓝条的颜色变为红色来表示。 红色虚线表示数据集中错误单元格的实际数量。 如图所示，大多数基于 ML 的方法和集合方法（包括 ED2、RAHA、Min-k (Min) 和 Max-entropy (Max)）的 F1 分数都在 0.92 和 0.99 之间，优于其他方法。 由于将数值属性转换为分类属性，一些检测器（如 NADEEF 和 KATARA）错误地将这些转换属性中所有干净的数值标记为噪声单元。 此类方法的精度较低（从 0.08 到 0.16 不等），通常会对修复阶段产生负面影响（参见第 6.3 节）。

图 2b 展示了应用于 Beers 数据集的检测器的 IoU 指标。 显然，基于 ML 的方法和集合方法具有很高的相似性（IoU 至少达到 98%）。 此外，图中还显示 NADEEF（F1 为 0.74）和元数据驱动（Meta，F1 为 0.48）方法的检测结果之间具有相对较高的相关性（IoU 为 87%）。 因此，我们可以推断出，元数据驱动方法的大多数检测结果（即 2570 个检测单元中的 2417 个）都违反了规则和模式。 同样，KATARA（F1 为 0.12）和 FAHES（F1 为 0.35）具有很高的相似性（IoU 为 88%），因为它们都采用了句法模式检测方法。 图 2c 描述了检测器的平均运行时间（对数刻度），红色条表示运行时间超过一分钟。 如图所示，由于需要搜索最佳配置、特征化和训练分类器，基于 ML 的方法需要较长的执行时间。 在检测同样的错误单元时，最大熵所需的时间比 ED2 少得多（至少少了 98%）（参见图 2b）。

图 2d 描述了使用七个检测器的引文数据集中检测到的细胞数量。 这样的数据集包含重复和错误标记的样本。 从图中可以看出，关键复制方法（DuplD）的表现优于其他所有方法，其平均 F1 得分为 0.86。 同样，集合方法（即 Min 和 Max）也比 Picket（基于 ML 的检测方法，平均 F1 分数为 0.18）取得了更好的性能（F1 分数介于 0.74 和 0.78 之间），这是因为 Picket 依靠自我监督来训练分类器，召回率较低。 此外，Clean- Lab 的 F1 分数较低，仅为 0.19，因为它只捕获了数据集中被错误标记的单元格，而忽略了重复的单元格。 图 2e 描述了密钥碰撞检测、ZeroER、Min-K 和 Max Entropy 之间的强 IoU 关系。 不过，ZeroER 需要更多时间（约两个数量级）来生成检测结果。

对于成人数据集，图 2f 显示了使用 11 个检测器检测到的细胞数量。 这样的数据集存在违反规则和异常值的问题，错误率很高。 在这种情况下，RAHA 和 ED2 的表现优于其他所有方法（平均 F1 分数分别为 0.8 和 0.78）。 根据它们的 IoU 值，HoloClean、NADEEF

和 Min-k 所获得的检测结果呈现出较高的相关性，这些方法只捕捉到了大部分违反规则的情况。 相反，dBoost 捕获了大部分异常值，却未能识别出违反规则的情况。 尽管 ED2 和 RAHA 能有效检测出该数据集中的错误单元，但它们的效率较低，平均需要 35 分钟才能找到错误单元，而 dBoost 和 Min-k 分别需要 2.3 分钟和 0.73 分钟。 智能工厂数据集是一个例子，说明相对较大的数据集存在明显的缺失值和异常值，错误率适中。 图 2h 显示了智能工厂数据集中使用 8 个检测器检测到的单元数量。 在这种情况下，Min-k 的表现优于其他检测器（平均 F1 得分为 0.75），同时所需的时间也比其他检测器少得多（参见图 2j）。 如图 2i 所示，RAHA 和 Meta 与 Min-k 的相关性相对较高。 此外，图 2h 显示 KATARA 产生了很多误报，这是因为它未能正确解释数据语义。

对于有回归任务的数据集，图 2k-2o 显示了各种检测器的检测精度和运行时间。 例如，图 2k 显示了在 Nasa 数据集中使用 12 个检测器检测到的细胞数量。 这样的数据集代表了受明确缺失值和异常值影响而误差率较小的小型数据集。 如图所示，最大熵和 dBoost 的表现优于所有其他方法（平均 F1 得分为 0.85）。 如图 2l 所示，在 IoU 指标为 0.99 的情况下，这两种检测器几乎产生了相同的检测结果。 尽管基于 ML 的方法检测到了大部分错误单元格，但由于存在大量误报，其 F1 分数介于 0.27 和 0.43 之间。 由于数据集较小，大多数检测器都能在一分钟内完成检测，如图 2m 所示。 至于自行车数据集，它有规则违反和异常值，错误率较小。 图 2n 显示了使用 11 个检测器在 Bikes 数据集中检测到的单元数量。 RAHA 和 Min-k 的平均 F1 分数分别为 0.72 和 0.75，优于其他检测器。 从图中可以看出，KATARA 和 NADEEF（平均 F1 分数分别为 0.25 和 0.4）的性能较差，因为它们会产生很多误报。 与 Nasa 数据集类似，dBoost 和 Max En- tropy 具有很高的相关性。 图 2o 显示，Min-k 比 RAHA 更高效，它平均需要 9 秒钟生成检测结果，而 RAHA 需要 6.6 分钟。

图 2p-2t 描述了使用与聚类任务相关的数据集时各种检测器的性能。 对于水数据集，它受到隐含缺失值和异常值的影响，错误率较小。 图 2p 显示，Max Entropy 和 RAHA 的准确率最高，平均 F1 分数分别为 0.74 和 0.76。 两种检测器的检测结果高度相关。 不过，与 RAHA（平均运行时间为 15.8 秒，标准偏差为 10.4）和 ED2（平均运行时间为 17.9 分钟）相比，Max Entropy 生成检测结果所需的时间要短得多（平均运行时间为 0.09 秒）。 RAHA 通常具有较高的方差，因为它在第一次迭代中需要花费相对较长的时间来创建用于生成训练特征的清洗策略。 对于 Power 数据集，NADEEF 和 Max Entropy 优于其他检测器，平均 F1 分数分别为 0.9 和 0.84，如图 2q 所示。 很明显，NADEEF 和 MVD 都具有很高的优先级。 不过，每种检测器都只捕捉到相关的误差。 换句话说，NADEEF 检测到了 1088 个模式违规（与错别字和隐式缺失值相关），而 MVD 只发现了显式缺失值。 在效率方面，Max Entropy 和 NADEEF 消耗的时间差不多（平均运行时间为 0.05 秒），而 ED2 平均需要 680 秒来生成检测结果。 对于 HAR 数据集，图 2r 显示 RAHA 的准确率最

高，平均 F1 得分为 0.89，但其检测耗时为 20.5 分钟（标准偏差为 20 分钟）（参见图 2t）。 图 2s 显示，MVD、HoloClean 和 Min-K 检测出了相同的缺失值错误单元格。

6.2.1 检测鲁棒性

在本节中，我们将研究各种误差检测器在准确性方面的鲁棒性。 为此，我们进行了两组实验，包括：(1) 改变数据集的错误率；(2) 改变离群值程度，离群值程度定义为偏离平均值的标准差数。 在前一组实验中，我们注入了离群值和缺失值，离群值程度设为 4。 在离群值程度实验中，我们注入了误差率为 30% 的离群值。 图 3a 比较了七个检测器在不同误差率下清理成人数据集时的鲁棒性。 很明显，所有检测器的 F1 分数在低误差率（即最高 0.02）时都呈线性增长。 在此范围内，几个检测器（如 ED2、最大熵和 Min-k）的斜率较大，这意味着检测精度较高。 当误差率进一步增大时，除 RAHA 外，大多数检测器的精度都会逐渐降低。 图 3b 显示了 Power 数据集上的类似实验。 如图所示，在低错误率情况下，ED2 比其他所有模型都获得了更高的准确率。 至于 RAHA，当错误率增加时，其性能也有所提高。 图 3c 比较了 10 个检测器在增加注入智能工厂数据集的离群值时的性能。 从图中可以看出，当离群值相对较小时（即低于 2），所有检测器的表现大致相同。 但是，当离群值超过 2 时，RAHA、ED2、Min-k、dBoost 和 Meta 的性能都有很大提高。

6.2.2 可扩展性分析。 在本节中，我们将评估几种误差检测器在处理大型数据集时的效率。 为此，我们进行了多次实验，以检测不同数据分数中的错误。 图 3d 和 3e 比较了十种检测器在处理足球数据集不同部分时的准确性和效率。 对于该数据集，图 3d 显示 ED2、NADEEF 和 RAHA 获得了最高的 F1 分数（即分别为 0.83、0.93 和 0.98）。 此外，该图还表明，有些检测器只能检测小部分数据。 例如，RAHA、ED2 在数据量为 50% 时停止工作，而 HoloClean 则在数据量为 90% 时终止工作。 图 3e 显示了平均运行时间的对比（对数标度）。 很明显，RAHA、ED2 和 KATARA 所需的时间（平均运行时间分别为 3.5、10.1 和 13.8 小时）远远超过其他探测器。 与 ED2 和 RAHA 相比，KATARA 能够检测出所有数据分数的错误。

6.3 数据修复

在本节中，我们将介绍根据各种错误检测器的检测结果生成修复候选数据的修复方法的结果。 我们根据每个数据集中的数据类型来划分实验。 此外，我们还介绍了面向 ML 的修复方法的结果，这些方法的输出是 ML 模型，而不是修复后的数据集。

6.3.1 分类属性。 图 4 显示了两个包含分类属性的数据集在修复精度和运行时间方面的修复结果。 例如，图 4a 显示了清理 Beers 数据集时的修复精度（精确度和召回率）。 从图中可以看出，如果采用最佳修复方法（由 GT 模拟）进行修复，RAHA、ED2、Min-k、Max Entropy、HoloClean 和 NADEEF 等多个检测器获得的检测结果都能带来较高的修复

准确率（平均 F1 分数为 0.99）。 如图 2a 所示，尽管 HoloClean 的召回率较低，但 HoloClean-GT 仍能达到较高的性能，因为 HoloClean 检测到了 254 个实际错误分类单元中的 248 个。 对于该数据集，BARAN 在为 RAHA、ED2 和 Max Entropy 的检测结果生成修复候选时，获得了最高的准确率（平均修复 F1 分数为 0.98）。 由于 KATARA 获得了大量假阴性结果（254 个错误分类单元中的 127 个单元）（参见图 2a），因此在使用地面实况进行修复时，最大修复 F1 分数仅为 0.66。 图 4b 比较了八种修复方法的运行时间。 方框内的蓝色区域代表特定点的运行时间标准偏差。 很明显，BARAN 消耗的时间（平均运行时间为 4.4 分钟，标准偏差为 1.5 分钟）远远超过其他所有检测器。

图 4c 显示了用于清理乳腺癌数据集的各种检测器/修复组合的修复精度。 如图所示，当使用 MissForest（F1 得分为 0.63）和 BARAN（F1 得分为 0.6）时，通过最大熵获得的检测结果具有中等精度。 此外，图中还显示，当使用地面实况对检测结果进行修复时，KATARA 的修复 F1 得分为 1。 事实上，KATARA 产生了很多假阳性（6843 个单元）和很少的假阴性（86 个单元，均为数值）。 因此，我们可以推断出，在存在高效修复方法的情况下，检测假阴性比检测假阳性对修复准确性的影响更大。 图 4d 显示，Holo Clean 和 BARAN 是最耗时的修复方法（平均运行时间分别为 45.7 秒和 53.8 秒）。

6.3.2 数值属性 图 5 用均方根值和运行时间描述了数值属性的修复结果。 例如，图 5a 比较了八种修复方法在清理智能工厂数据集时的性能。 每种修复方法由一组条形图组成，代表不同的检测方法。 红色虚线表示脏版数据集的 RMSE 值。 图中显示，在不同的修复方法中，RAHA 和 dBoost 的检测性能最高（RAHA 和 dBoost 的平均 RMSE 分别为 0.93 和 0.82）。 此外，图中还显示 GT 可生成 RMSE 与脏版本相当的修复版本（参见 GT 组中 FAHES、Meta 和 NADEEF 的条形图）。 这样的修复性能通常是由于这些检测的准确性较低造成的。 因此，我们可以得出这样的结论：如果没有准确的错误检测过程，高效的修复方法也会效果不佳。 图 5c 显示，在乳腺癌数据集中，ED2 和 RAHA 的检测结果在大多数修复方法中达到了最高的修复精度。

对于自行车数据集，图 5d 显示，大多数清理策略生成的修复版本都相对优于脏数据。 不过，FAHES、HoloClean 和 KATARA 的检测结果所生成的修复版本的 RMSE 值要高于脏数据版本（参见虚线上方标准和基于 ML 的估算方法的条形图）。 对于该数据集，BARAN 所需的时间（平均运行时间为 58.4 ± 40.2 分钟）远高于所有其他方法。 图 5e 比较了十种修复方法在清理 Water 数据集时的准确性。 从图中可以看出，所有修复版本的性能都与脏版本相近或更好。 显然，在所有修复方法中，RAHA 和 Max Entropy 的准确率最高（平均 RMSE 分别为 0.7 和 0.65）。 在运行时间方面，图 5f 显示 HoloClean 是最耗时的方法，平均运行时间为 5.2 ± 4 分钟。

6.3.3 面向 ML 的修复方法 本节将介绍 ActiveClean、CPClean 和 BoostClean 等面向 ML 的方法的结果。 图 6 比较了这些方法在建模精度方面的性能。 其中，图 6a 显示了在成人数

据集的场景 S1、S4 和 S5 中生成模型的 F1 分数。 从图中可以看出 对于乳腺癌数据集，图 6b 显示在 S1 中由 Active- Clean 生成的模型普遍存在准确率低的问题，而在 S4 中的平均 F1 分数比 S1 高出约 88%。 造成这一结果的主要原因是数据集规模较小，而且所有检测器的检测准确率都相对较低（即最大熵的最高 F1 得分为 0.75）。 就 CPClean 和 BoostClean 而言，三种情况下的结果彼此接近。

6.4 建模精度

在本节中，我们将介绍在不同场景下对各种数据集建模的结果。 图 7 展示了在不同数据版本上训练的不同分类、回归和聚类模型的准确性。 对于 Beers 数据集，图 7a 显示了在场景 S1 和 S4 中六个分类器的平均 F1 分数。 如图所示，所有分类器的性能都随修复数据的质量而变化。 例如，MLP 分类器在 S4 中的平均 F1 得分为 0.732，而在 S1 中的准确率在 0.368 到 0.727 之间。 图 7b 通过比较 S1（蓝色）和 S4（绿色）中不同版本的 Beers 数据集（即脏数据集 (D0)、地面实况数据集和修复数据集）上训练的 MLP 模型的性能，阐明了这些结果。 很明显，蓝色和绿色区域大多相互重叠。 唯一例外的是 X3 组合，代表最大熵和标准估算。 在多个分类器中重复出现的这种低准确率，通常是由于不同标准归因方法产生的低质量修复造成的。 图中，A/B 统计检验的结果以蓝色填充/空方形标记的形式表示。 在这种情况下，填充标记表示可以拒绝零假设$H0$（即 S1 和 S4 中的两个 MLP 模型是不同的），而空标记则表示无法拒绝$H0$。 因此，我们可以确认，如果我们运行十次以上的实验，S1 和 S4 中模型的性能差异将保持不变。 图 7c 比较了在不同版本的成人数据集上训练的十个分类器的结果。 在这张图中，S1 中结果的分布情况使我们能够确定哪些 ML 模型对数据质量问题具有鲁棒性。 例如，S1 中 DT 的结果范围在 0.17 到 0.99 之间，而 Ridge 的结果范围在 0.74 到 0.78 之间。 图 7d 显示了在不同版本的成人数据集上训练 SVC 时的性能。 对于大多数数据版本，SVC 在两种情况下的准确率相当。 尽管 ED2 的检测准确率很高，但在其大多数修复版本（如 E1、E3、E10 和 E15）中，ED2 的检测都出现了质量问题。 出现这种情况的原因是 ED2 产生了大量的误报（118,741 个单元）（参见图 2f）。 同样，图 7e 和图 7f 描述了不同版本的乳腺癌数据集的建模准确性。 对于该数据集，DT 表现良好，F1 分数范围在 0.65 到 0.94 之间，而 GNB 的 F1 分数范围在 0.15 到 0.85 之间。 图 7f 显示，在乳腺癌数据集的大多数修复版本中，XGBoost 在 S4 中的性能略优于 S1。

对于包含重复和错误信息的引用数据集，图 7g 展示了多个分类模型在 S1 和 S4 情景下的 F1 分数。 从图的右上角可以看出，在采用 "删除 "策略时，大多数分类器的性能都与基本事实相似。 其他依赖基于 ML 估算的清理策略，如 M6、M7、M9、X7、X6 和 X9，会导致预测性能大幅下降（参见图 7h）。 与其他分类器不同，XGBoost 在脏数据和修复程度最高的数据版本中表现不佳（如图 7g 所示，F1 分数在 0.05 到 0.8 之间波动，并且在 0.26 值以下具有较高的密度）。 为了进一步了解错误标签的影响，我们在成人和乳腺癌数据集

上进行了实验，在标签中添加了噪声（即翻转一些二进制标签）。 实验结果表明，在脏数据集上训练的 MLP、RF、DT 等多个 ML 模型的性能略低于在真实数据集上训练的相同模型（对于 RF，脏数据集的平均 F1 得分为 0.9，而真实数据集的平均 F1 得分为 0.93）。

图 7j-7o 展示了在不同数据集上训练的各种回归模型的性能。 如图 7j 所示，XGB 在 S4 中的准确率最高（RMSE 为 1.54）。 不过，它的性能在很大程度上取决于修复的质量（参见 S1 中的 RMSE 值，从 1.78 到 35.9 不等）。 相反，在 S1 中，DT 和 RF 的 RMSE 值分布较窄。 图 7k 显示，DT 对修复最多的数据版本具有大致相同的预测性能。 图中还显示了一些清理策略，例如 X2、X7、X8、N11 和 K11，它们的性能与地面实况相似。 对于土壤湿度数据集，图 7l 显示，在 S1 中 RMSE 值分布相对紧密的情况下，KNN 的性能优于其他模型。 对于该数据集，如图 7m 所示，使用地面实况修复 RAHA 的检测结果与 S4 中获得的 RMSE 值相当。 在图 7n 和 7o 中，我们展示了 RANSAC 和贝叶斯 Ridge 在 S2 和 S3 场景中的性能（参见表 3）。 很明显，RANSAC 和 Bayesian Ridge 在 S2 中的表现比在 S3 中要好得多。 由于这一结果出现在所有其他数据集上，我们可以推断出，在脏数据或质量相对较低的修复数据上训练出来的模型，在使用高质量数据进行测试/服务时可能会表现良好。

除了回归，我们还根据剪影指数测量了几种聚类方法的准确性，如图 7p-7t 所示。 结果显示，一些聚类方法（如 Optics、GMM 和 HC）在 S1 和 S4 中的性能相当，甚至在 S1 中几个修复版本的性能更好，如图 7p 和 7r 所示。 例如，图 7q 比较了 Birch 对不同版本的水数据集进行聚类时的性能。 总体而言，Birch 在 S4 中的表现优于 S1。 不过，有几种经过修复的方法的聚类性能比地面实况更好（R1、R7 和 R9 的平均性能分别提高了 16%、18% 和 17%）。 图 7s 显示了 K-Means 对电力数据集进行聚类时的类似结果。 最后，图 7t 比较了在 HAR 数据集上训练的五种聚类方法的性能。 图中显示，所有模型在 S1 中的分布都相对紧密，这意味着对修复版本的质量不敏感。 使用 RAHA 检测生成的几个修复版本（如 R1、R2 和 R6）的性能与地面实况相似。

6.5 经验教训

主要发现： 在本节中，我们将重点介绍本研究的主要发现和经验教训。 通过大量实验，REIN 证明了脱离下游应用（如预测任务）孤立地评估错误检测和修复方法会产生广泛的误导。 例如，图 2a、2h 和 2n 显示，KATARA 存在许多误报。 此外，为 KATARA 分段生成的修复质量有时还不如数据集的脏版本（参见图 5d）。 不过，图 7d、7g、7i 和 7k 清楚地表明，基于 KATARA 修复数据版本训练的 ML 模型的预测性能与其他模型相当。 对于其他检测器，如 FAHES、NADEEF 和 HoloClean，也可以得出类似的结论。 事实上，大多数错误检测和修复方法通常都是利用其相对于地面实况的性能来进行评估的 [20, 32, 33, 38, 46]。 因此，上述发现是一项重要成果，可指导研究人员和开发人员如何有效评估数据清理方法。

另一个有趣的发现是，与回归模型和分类方法相比，分类模型对属性错误具有更强的鲁棒性。 通过比较图 7 中不同模型的性能，我们可以清楚地看到，几乎所有分类器的 S1（蓝色区域）和 S4（绿色区域）之间的差异都非常小。 相反，回归模型和聚类方法在 S4 中的表现明显优于 S1。 因此，数据清理是回归和聚类应用管道的必要组成部分。 此外，分类应用可能不需要实施复杂的数据清理方法。 简单的清理方法就能为分类模型提供训练模型所需的必要质量水平。 同时，简单的错误检测和修复方法不需要过多的时间，因此我们可以大大加快数据准备过程。 在存在类错误的情况下，一些分类器的性能相对较差。 因此，有必要采用自动错误标签检测方法来生成准确的预测结果。对于所研究的 AutoML 算法，即 TPOT 和 Auto-Sklearn，结果显示它们并不总是能生成最准确的模型。 例如，在乳腺癌数据集中，使用 X13 和 X15 的 TPOT 生成的模型的 F1 分数分别为 0.75 和 0.6。 相比之下，使用 B15 和 X2 的 TPOT 的 F1 分数分别约为 0.98 和 0.99。 因此，在数据清理不当的情况下，这些算法可能无法生成准确的模型。

错误检测器： 关于错误检测方法，如图 2 所示，在大多数情况下，基于 ML 的方法和集合方法的检测准确率高于其他非学习方法。 不过，结果也表明，大多数检测器在不同数据集上缺乏一致性，即在不同数据集上的表现各不相同。 例如，图 2a 显示，在 Beers 数据集中，ED2 能高精度地检测出所有错误。 然而，它在其他数据集（如 Adult、Nasa 和 HAR）中也出现了误报和误否（参见图 2f、2k 和 2r）。 同样，NADEEF 在 Nasa 数据集上的表现不佳（平均 F1 得分为 0.12），而在 Power 数据集上的表现尚可（平均 F1 得分为 0.91）。 基于 ML 的检测器的其他缺点如下： (1) 它们无法识别错误类型，也就是说，它们只能对每个单元格是否出错做出二元判定。 这种行为可能会使选择合适的数据修复工具变得复杂。 (2) 可扩展性差（参见图 3d-3e 中的结果）。 (3) 它们需要用户干预来标注数据。 因此，为了解决上述缺点，有必要加大力度推进基于 ML 的检测器。

结果表明，基于规则的 er- ror 检测器的性能主要取决于用户提供的规则/约束的数量和质量。 例如，在成人数据集中，当提供的规则数量从 17 条减少到 7 条时，HoloClean 的 F1 分数从 0.51 降到 0.12。 因此，将自动规则/限制条件生成器与此类检测器整合在一起以提高其性能至关重要。 在这方面，我们强调无配置方法通常简单易用，但通常需要很长时间才能找到最合适的配置，例如 dBoost 和 RAHA（参见图 2c、2j 和 2t）。 值得一提的是，目前的 RAHA、ED2 和 Meta 在脏数据集中存在重复数据的情况下无法工作。 出现这一问题的主要原因是脏数据集和地面实况数据集的长度不同。 在这种情况下，这些检测器就无法使用地面实况来模拟人类标注者，即标注脏单元格。 Picket 是一个例外，因为它依赖于自我监督。 因此，它并不强制要求用户提供标签。 然而，研究结果表明，Picket 只适用于小型数据集，由于自我监督的复杂性，它不能很好地扩展。 对于较大的数据集，如《成人》和《智能工厂》，Picket 由于会导致内存故障而被终止。

修复方法： 为了获得更好的修复体验，我们发现检测精度对修复质量的影响相对高于检测召回率（参见图 2a 和 2n）。 这种优势背后的原因是为了避免误报，因为误报可能会促使所采用的修复方法引入新的错误单元格或删除所有检测到的单元格，从而导致修复后的数据集与地面实况完全不同步。 然而，有效的修复方法甚至可以避免检测阶段误报的负面影响。 例如，在 Beers 数据集中，NADEEF 就产生了许多误报。 然而，在使用 GT（模拟高效修复方法）修复检测时，这些误报几乎没有影响。 在这种情况下，如果使用高效修复方法，检测阶段的假阴性比假阳性的危害更大。

对于面向 ML 的修复方法，我们注意到 CPClean 和 BoostClean 几乎不适用于与多类分类任务相关的数据集。 其根本原因在于，这些方法将每个脏数据集分成若干批次，而每个批次必须包含所有类别的样本。 然而，当存在多个少数类别时，从每个类别中获取样本并不总是可能的。 对于二元分类问题的数据集，如果标签中包含错误的单元格，CP-Clean 和 BoostClean 可能会因为在标签中引入新值而失效，从而将问题转化为多类分类。 就 ActiveClean 而言，它首先要对脏数据集进行分区，以获得干净的部分（即没有任何错误的数据部分）用于预热。 这样的分区需要代表数据集中所有可能的类别。 因此，ActiveClean 会搜索满足此条件的分区。 如果找不到这样的分区，就会返回异常。 这种问题可能发生在以下情况：(1) 数据集中有太多具有多个次要类别的类别（如啤酒）；(2) 数据集中没有足够的清洁单元格。

可行建议： 基于在 REIN 中获得的结果，我们在设计或选择数据清理工具时提出以下建议：(1) 根据计划的下游应用设计和评估数据清理方法，以正确选择适合的清理方法；(2) 在分类任务中采用简单的清理策略（非学习检测器和通用修复方法）来消除属性错误，在回归和聚类任务中采用更高级的清理方法（基于 ML）；(3) 利用高级技术来消除类错误，例如，CleanLab、数据估价、标签修复等、 清洁实验室、数据估值、标签平滑和噪声感知学习 [43，50]；(4) 采用自动化工具，如 (4) 采用自动化工具，如 FDX 剖析器和 Metanome [41]，提取完整性约束和功能依赖规则，以正确使用清理工具，如 NADEEF 和 HoloClean，尽量减少用户参与；(5) 采用重复检测工具，如 (6) 在准备大量数据（如图 3d 所示，超过 50k 行）时，避免使用基于 ML 的错误检测器，如 ED2、RAHA 和 Picket，因为它们的可扩展性较差。

7 相关研究

事实上，对现有数据清理方法进行调查或比较的研究很少[2, 28, 29, 47]。 Lee 等人[28]对五种数据清理方法进行了调查，并提出了几个研究方向，如将数据清理方法与可视化界面相结合、使用高性能内存管理硬件解决方案等。 同样，Ridzuan 等人[47] 综述了数据清理方法及其在处理大数据时面临的挑战。 CleanML[29]介绍了一种关系数据库模式，旨在组织研究数据清洗对 MLclassification 任务影响的实验结果。 由于不考虑每个数据集的地面实况，CleanML 忽略了比较使用地面实况和修复数据集训练的 ML 模型的性能。 此外，

CleanML 将评估局限于简单的分类任务，而忽略了其他 ML 任务，如回归、聚类和 AutoML 算法。 此外，CleanML 没有考虑整体、半监督或面向 ML 的错误检测和修复方法。 在 REIN 中，我们解决了这些不足，将我们的发现推广到实践者和数据科学家，为他们处理表格数据中的数据清理问题提供正确的指导。

8 结论

在本研究中，我们引入了一个名为 REIN 的基准框架，用于正确评估错误检测和修复方法。 REIN 使 ML 工程师和从业人员能够在 ML 管道中选择最合适的数据清理方法。 我们进行了广泛的实验研究，涉及 19 种检测器、19 种修复方法、33 种 ML 模型和 14 个数据集。 研究结果表明，脱离下游应用来评估数据清洗方法会产生广泛的误导。