AutoCure：面向机器学习流水线的自动化表格数据整理技术

Mohamed Abdelaal*Software AG*德国达姆施塔特
moha
med.abdelaal@softwareag.com

Rashmi Koparde马格德堡奥托·冯·格里克大学德国马格德堡
rashmi.koparde@st.ovgu.de

Harald Sch¨oning*Software AG*德国达姆施塔特
harald.schoening
@softwareag.com

arXiv:2304.13636v1 [cs.DB] 26 Apr 2023

摘要——**机器学习算法已日益广泛地应用于自动驾驶、医疗保健和金融等多个领域。在这些领域中，数据准备仍是开发准确模型的一项重大挑战，因为要在庞大的搜索空间中寻找合适的数据整理与转换工具，需要投入大量专业知识和时间。为应对这一挑战，我们提出了 AutoCure，这是一种新颖且无需配置的数据整理流水线，可提升表格数据的质量。与传统数据整理方法不同，AutoCure 通过自适应集成式错误检测方法和数据增强模块，以合成方式提高干净数据比例。在实践中，AutoCure 可与 Auto-sklearn、H2O 和 TPOT 等开源工具集成，以推动机器学习民主化。作为概念验证，我们将 AutoCure 与 28 种传统数据整理工具组合进行了比较评估，结果表明，无需用户干预，AutoCure 也能取得更优的性能和预测准确率。我们的评估显示，AutoCure 是一种有效的方法，能够自动化数据准备流程并提高机器学习模型的准确率。**

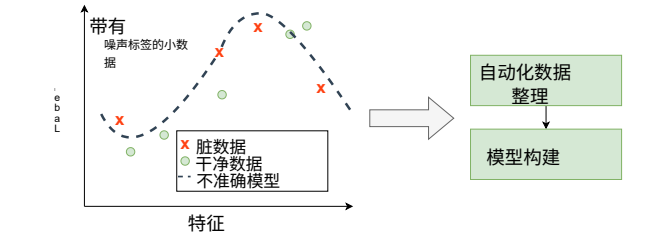
索引词—**数据整理、数据质量、数据增强、机器学习、表格数据**

I. 引言

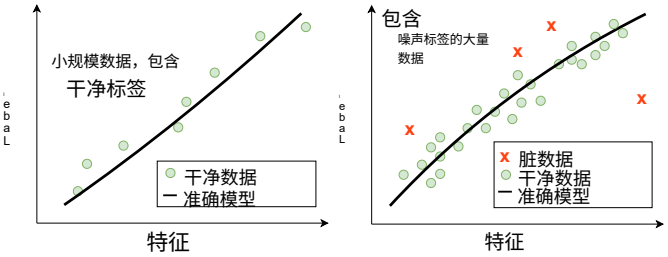
数据质量问题： 近几十年来，机器学习（ML）在多个应用领域产生了深远影响，包括自动驾驶、游戏、医疗保健、物流和金融 [9]。在这些领域开发机器学习模型通常需要从多个不同来源获取数据。例如，自动驾驶汽车中的感知和状态估计系统通常使用基于传感器读数、GPS定位点和历史记录所收集数据训练的机器学习模型 [35]。在完成数据获取后，通常会执行数据准备与转换过程，使所收集的数据达到适合模型构建的状态。事实上，此类机器学习模型的性能在很大程度上取决于输入数据的质量 [25]。具体而言，只要输入数据包含噪声或错误实例，预测性能就可能大幅下降。总体而言，由于不当的连接操作、噪声通信信道、不准确和/或不完整的手动数据录入等原因，现实世界的数据大多具有异构错误特征 [16]。由于这些技术性和人为相关问题，不同的错误类型，例如离群值、模式/规则违反、重复项、不一致-性以及隐式/显式缺失值，可能会同时出现在一个数据集中。

挑战： 由于高数据质量是提升机器学习模型性能的必要前提，因此在用于建模任务之前，必须妥善整理输入数据。典型的数据整理流水线通常从检测错误实例开始，然后对其进行修复。事实上，已有大量用于错误检测和修复的商业及开源工具，例如 Talend、OpenRefine、Trifacta、Tamr、NADEEF 和 HoloClean [2]。然而，这些工具仍存在若干问题。首先，它们通常需要领域知识，以及能够将此类知识表述为一组规则/约束的专业人员。违反这些规则/约束的情况通常会被报告给用户，以便用户采取适当的纠正措施，例如修正数据，或微调数据和/或规则定义。由于需要用户参与，若用户不具备足够的数据专业知识，采用这些数据整理工具可能会颇具挑战。一种可能的解决方案是采用自动规则生成工具，例如 Metanome [27], DCFinder[28],和 RTClean [11]。然而，根据我们的实验[1]，此类工具的性能（以生成规则的质量衡量）在不同数据集之间差异很大。此外，其较高的计算复杂度也阻碍了它们与数据整理工具的集成。

其次，大多数基于机器学习的错误检测工具，例如 RAHA[23] 和 HoloDetect [15], ，无法识别检测到的错误类型，即无法判断错误属于规则/模式违反、离群值、缺失值还是重复项。它们只是训练一个检测分类器，用于区分错误数据实例和干净数据实例。由于缺乏关于检测到的错误类型的信息，选择合适的数据修复工具并非易事。一种权宜之计是实现多个数据修复工具。然而，这种方案可能会增加数据整理流程的复杂性，因为修复候选项的搜索空间会大幅扩大。例如，如果某个数据实例 $X_i = x_{i,1}, \cdots, x_{i,k}$ 中存在一个空单元格 $x_{i,j} = NaN$，其中 k 是列数，那么就必须对单元格 $x_{i,j}$ 进行适当的插补。鉴于存在大量能够生成不同值的插补方法 $a_1, \cdots, a_n$，选择最佳修复候选项便颇具挑战。为此，有必要进一步扩展数据准备流程，而这反过来



（a）噪声对模型构建的影响



（b）传统数据整理　　（c）AutoCure

图 1. 在不同类型数据上训练的三个机器学习模型

会通过实现额外的工具（例如 BoostClean [20] 和 CPClean [17]）增加复杂性。

所提方法： 为应对这些挑战，我们提出了一种新颖的自动化数据整理流水线，称为 AutoCure。该流水线由两个主要模块组成，即自适应集成式错误检测模块和干净数据增强 模块。AutoCure 背后的核心思想是，以合成方式提高输入数据中干净子集的密度（输入数据通常由干净数据实例和脏数据实例组成）。根据信息论，噪声降低的过程等价于增加更多质量相近的数据 [32]。基于这一理论，AutoCure 致力于提高干净数据实例的比例，以降低噪声/错误数据实例的影响。因此，我们可以通过干净数据增强模块替代数据修复，从而避免数据修复带来的问题。

为进一步阐明 AutoCure 背后的理念，图 1 展示了三个不同场景的示例。图 1a 展示了一个在小型数据集上训练的机器学习模型（虚线曲线），该数据集包含一组干净实例（绿色圆圈）和一组含噪数据实例（红色圆圈）。由于存在含噪实例，这样的机器学习模型无法准确描述线性数据集。为提高预测性能，我们可以使用传统数据修复方法修复含噪数据实例，如图 1b 所示。然而，修复过程通常较为复杂，需要用户干预，并且还需采取额外步骤来选择最佳修复候选项。此外，传统数据修复方法无法还原含噪实例的真实值。AutoCure 采用的另一种策略是有意提高干净实例的密度，以应对数据质量不佳的问题，如图 1c 所示。在这种情况下，脏实例对机器学习建模过程的影响会显著降低。因此，生成的机器学习模型整体上与图 1b 所示的曲线相似。

贡献总结： 总而言之，本文的贡献如下：（1）我们提出了一种新颖的两阶段数据整理流水线，通过提高干净数据实例的密度来处理错误表格数据。（2）我们提出了一种自适应集成式错误检测方法。如果某个单元格至少被 k 个基础检测器检测到，所提出的检测器就会将该单元格 $x_{i,\ j}$ 标记为脏数据。AutoCure 会动态调整阈值 k，以克服从输入数据集中提取干净数据实例时通常会出现的数据排除问题。这类数据问题通常是由于基础检测器的检测不准确所致（参见第三节）。（3）我们进行了大量实验，根据真实值评估 AutoCure，以获得性能上界，并将其与广泛的基线方法在建模准确率和训练时间方面进行比较。结果表明，AutoCure 始终能达到与真实值数据相当的性能，同时，与运行多种修复方法以选出最佳修复候选项的做法相比，所需训练时间更短。据我们所知，AutoCure 是首个通过降低错误数据的影响，并将干净子集的密度视为解决数据质量问题的实用方案，从而在机器学习流水线中自动进行数据整理的工作。

论文结构： 本文其余部分的结构如下。第二节介绍所提流水线的不同组成部分，并重点说明主要假设。第三节介绍自适应集成式错误检测方法，并解释数据排除问题。在第四节中，我们讨论使用所谓的变分自编码器（VAE）进行干净数据增强。第五节介绍我们的概念验证实现，并展示针对不同数据集所获得的结果。第六节讨论相关工作，最后第七节总结全文并展望未来工作。

II. 概述与架构

本节将介绍 AutoCure 的架构及我们的假设。所提出的架构由两个模块组成：自适应集成式错误检测模块和干净数据增强模块，分别执行三项主要任务，包括：（1）检测错误数据实例；（2）提取包含所有类别的干净子集；（3）从与干净子集具有相同分布的数据中生成附加数据。图2概述了所有组件及其相应输入。首先，将一个脏数据集作为自适应集成式错误检测模块的输入。该模块旨在最大化检测召回率，即被检测出的错误数据实例占错误数据实例总数的比例。此外，这种自适应方法使我们能够应对数据排除问题（参见第三节），而这些问题通常是由较差的检测性能引起的。

基于自适应集成的错误检测模块实现了多种基础错误检测器，例如缺失值（MV）检测器、离群值检测器、重复值检测器，

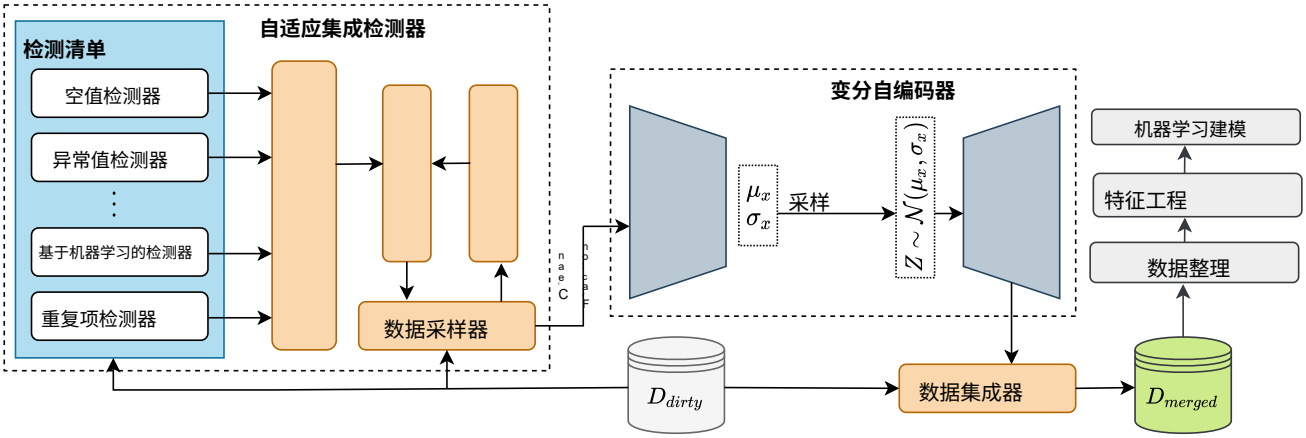


图 2. 所提方法的架构

规则违规检测器和基于机器学习的错误检测器（更多详情请参见第五节）。每个基础检测器都会生成一个索引列表，列出该方法检测到的错误实例。为整合这些检测结果，我们实现了一种自适应投票机制，将至少被 k 种方法检测到的实例标记为错误实例。生成错误数据实例列表后，使用数据采样器 提取干净数据比例。在扩充干净数据比例之前，数据检查器 会仔细检查该比例，以检测是否可能出现相关的数据排除问题。如果检测到其中一个问题，数据采样器便会迭代调整投票阈值 k，直到解决所有相关的数据排除问题。

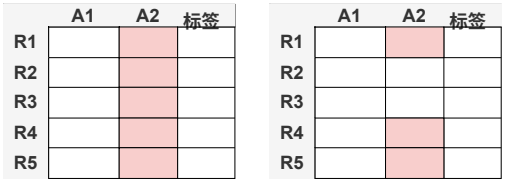
提取出的干净数据比例随后作为变分自编码器 （VAE）的输入，用于从相同分布中生成一组新的数据。如图2所示，VAE模块由两个前馈神经网络构成，即编码器和解码器。编码器学习干净数据比例的潜在空间表示的分布。随后，解码器使用采样得到的潜向量生成与输入相似的数据实例。在此情境下，优化问题是最小化（1）重构损失函数，该函数比较输入与解码器生成的值；以及（2）KL散度，该散度从统计角度区分输入数据与生成数据的概率分布。生成额外的干净数据后，数据集成器 会将新生成的数据与原始脏数据合并。随后，在使用转换后的数据训练机器学习模型之前，通过特征工程和数据整理工具（例如归一化、嵌入和特征交叉）对合并后的数据集进行转换。在本文中，我们假设脏数据集具有异构错误特征。换言之，一个数据集中可能同时存在不同类型的错误，例如离群值和缺失值。为了实现干净数据增强，我们还假设每个脏数据集都包含一组干净实例和一组脏实例。下一节将介绍自适应集成式错误检测模块的实现细节。

III. 自适应错误检测

在深入探讨自适应集成式错误检测方法的细节之前，有必要指出，现有的依赖错误 检测器只能处理数据集中的一部分错误。例如，缺失值检测器只能找出空值，却会忽略其他错误。类似地，离群值检测器通常采用统计度量来区分合法数据与离群值，却忽略规则违反和重复项等其他错误类型。因此，这类检测器较低的检测召回率使其无法单独使用。为了最大限度地提高检测召回率，已有若干先进方法，可归为两大类别，即基于机器学习的方法 和集成方法。第一类包含采用半监督二元分类器来区分干净数据实例和脏数据实例的检测器 [15], [26]。第二类包含结合多个依赖错误的方法的检测结果的检测器 [2]，既能提高检测召回率，也能提供一个用于控制检测精确率的调节参数（检测精确率定义为检测出的实例中相关实例所占的比例，例如实际脏数据实例所占比例）。

然而，当主要目标是从脏数据集中提取干净子集（这是 AutoCure 的要求）时，基于机器学习的检测器和集成检测器通常会受到相关数据排除问题的影响。具体而言，当提取的干净子集出现以下情况之一时，通常会产生数据排除问题：(1) 为空；或 (2) 未包含脏数据集中存在的所有类别（即数据集中的唯一标签）。第一种类型称为属性级 排除，通常发生在错误检测器将某个属性的所有数据实例标记为错误时（参见图 3a 中属性 $A2$ 的红色实例）。在这种情况下，由于所有记录都包含错误数据实例，数据采样器将无法提取干净子集。第二种类型称为类别级 排除，发生在错误检测器将一个或多个类别中的所有数据实例标记为错误时（参见图 3b 中的红色实例）。因此，这些类别（例如图 3b 中的类别“1”）将不会出现在干净子集中，而这

反过来会大幅改变增强数据的数据分布。具体而言，这两种数据排除问题可能源于：(1) 某些依赖错误的检测器或基于机器学习的检测器产生的假阳性；以及 (2) 在集成检测器中为所有数据实例固定投票阈值 k。



(a) 属性级　　(b) 类别级

图 3. 数据排除问题示例

为解决数据排除问题，AutoCure 引入了一种自适应集成方法，可动态调整投票阈值，以保留提取出的干净子集中的所有类别。为此，所提出的基于自适应集成的方法会迭代地作出检测决策。具体而言，每当数据采样器检测到数据排除问题时，就会调整投票阈值 k 的值。图 4 展示了一个示例：在脏玩具数据集中检测错误时，如何调整投票阈值 k 。该数据集包含五条记录 $R1$-$R5$ 和两个属性$A1$ 和 $A2$。运行所有可用的检测器（即 $S1$-$S7$）后，迭代投票机制会从投票阈值的初始值开始，即 $k=3$。在此情形下，单元格 $C12$已被三种检测方法检测到，即 s_1、s_3 和s_4；而单元格 $C42$ 已被四种检测方法检测到，即 s_1、s_2、 s_5 和 s_6。

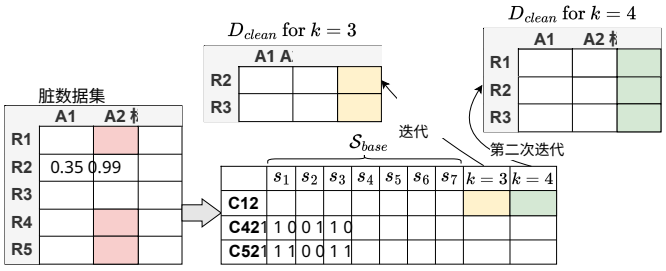


图 4. 调整投票阈值的示例

根据传统的 Min-K 投票机制，单元格 $C12$ 和 $C42$ 被标记为错误，因为至少有三种方法检测到了它们。因此，包含这些单元格的记录将从干净子集中排除，如左上方的表格所示（即， D_{clean}对于 $k=3$）。由此产生了类别级数据排除问题：干净子集中缺少类别为“1”的记录。为解决这一问题，AutoCure 在第二轮迭代中将投票阈值 k 从三提高到四。在这种情况下，单元格 $C12$ 未超过阈值。因此，它被标记为干净单元格，其所在记录也将被纳入干净子集中，如右上方的表格所示（即，D_{clean} 对于 $k=4$）。

为阐明 AutoCure 的实现细节，图 5 展示了所提出的数据整理流水线的伪代码。首先，AutoCure 汇总了若干依赖错误的检测器和基于机器学习的检测器 $\mathcal{S}_{base}$。每个检测器 s_i 都会生成错误实例的索引列表，例如 $R_i=\{(r_u,\ a_v)\}$，其中 r_u、 a_v 分别表示脏数据集中的第 u 条记录和第 v 个属性 D_{dirty}。随后，在估计单元格计数 Q_{cells}之前，会合并所有 m 检测器 R_{all} 的检测结果；该计数用于量化每个单元格 c_j被检测出的次数（参见第 5-7 行）。起初，AutoCure 将当前投票阈值 k_{cur} 设为一个大于 1 的较小值（例如，$k_{cur}=2$ ，而设置 $k_{cur}=1$ 则意味着仅使用一种依赖错误的检测方法或基于机器学习的检测方法）。在第一次迭代中，算法使用初始值，即 $k_{cur}=k_{init}$，来判断每个单元格是否存在错误（参见第 16 和 17 行）。随后，利用最终的检测结果列表 $R_{ensemble}$ 提取干净子集 D_{clean}，如第 18 行所示。

在运行变分自编码器之前，该算法会检查是否存在数据排除问题（参见第19—27行）。如果干净数据比例的大小等于零，则可检测出属性级数据问题。如果检测到属性级问题，算法便会将临时阈值 K_{attr} 提高更新率 α_u 所对应的值（参见第20行）。提高阈值显然意味着放宽投票机制，进而减少检测出的单元格数量。在后续迭代中，算法会重新估计所有检测结果的列表 R_{all} 以及干净子集 D_{clean}。接下来，它会再次检查是否存在潜在的属性级数据排除问题，然后检查 D_{clean} 是否存在类别级排除问题。为此，它只需比较脏数据中的类别 类别(D_{dirty})数量与干净子集中的类别 类别(D_{clean})数量。如果检测到类别级问题，AutoCure 会估计缺失类别的列表 L_{miss}（参见第22—23行）。在这种情况下，AutoCure 会有意区分属于其类别包含在 L_{miss} 中的记录的单元格，例如图4中的 $C12$ 和 $C42$ ，以及所有其他单元格，例如 $C52$。

这种区分背后的依据是，在确保 D_{dirty} 中的所有类别也存在于干净子集 D_{clean} 中的同时，实现较高的检测精确率。为此，AutoCure 放宽了对属于缺失类别的单元格的投票算法，例如 $C12$ 和 $C42$，以便将它们纳入干净子集。相比之下，AutoCure 对其他单元格（例如 $C52$）的要求更严格，因为这些单元格所属的类别已在干净子集中有所体现。为此，AutoCure 采用两个不同的临时阈值 k_{attr} 和 k_{class}。发生类别级排除时，它会更新阈值 k_{class} 。在下一轮迭代中，算法会检查每个单元格 c_j 是否属于缺失类别的记录（参见第 12 行）。如果是，则使用临时阈值 k_{class} 进行检测决策，该阈值始终比 k_{attr} 高出更新率 α_u 的数值。否则，它会使用阈值 k_{attr}。算法会持续迭代，直到

要求：脏数据集 D_{dirty}、初始投票阈值 k_{init}、增强预算 N_{aug}、基础检测器 $\mathcal{S}_{base}$ $=s_1$、$\cdots$、s_m、更新率 α_u 1：对所有检测器 $s_i \in \mathcal{S}_{base}$ 执行 ▷ 运行所有可用检测器 2：估计检测结果 $R_i \leftarrow s_i(D_{dirty})$ 3：将 R_i 添加到所有检测结果列表 R_{all}4：初始化单元格计数器 $Q_{cells} \leftarrow 0$5：对所有单元格 $c_j \in R_{all}$ 执行 ▷ 估计单元格计数 6：如果单元格 c_j 在单元格计数器 Q_{cells} 中，则 7：递增计数器 $Q_{cells}[c_j] \leftarrow Q_{cells}[c_j] + 1$8：初始化缺失类别列表 $L_{miss} \leftarrow \emptyset$9：初始化临时阈值 k_{attr}、$k_{class} \leftarrow k_{init}$10：当条件为真时执行 11：对所有单元格 $c_j \in Q_{cells}$ 执行 ▷ 自适应最小 K 投票 12：如果 classes(c_j) $\in L_{miss}$ 则 13：设置当前阈值 $k_{cur} \leftarrow k_{class}$14：否则 15：设置当前阈值 $k_{cur} \leftarrow k_{attr}$16：如果 $Q_{cells}[c_j] \geq k_{cur}$ 则 17：将单元格 c_j 添加到最终检测结果列表 $R_{ensemble}$18：提取干净子集 $D_{clean} \leftarrow D_{dirty} \backslash R_{ensemble}$19：如果 $D_{clean} = \emptyset$ 则 ▷ 属性级数据排除 20：更新阈值 $k_{attr} = k_{attr} + \alpha_u$21：否则，如果 classes($D_{clean}$) $\neq$ classes(D_{dirty})，则 ▷类别级 22：查找 $L_{miss} \leftarrow$ 类别 D_{dirty} \ 类别 D_{clean}23：更新阈值 $k_{class} = k_{class} + \alpha_u$ 24：如果差值 $|K_{class} - k_{attr}| \geq 2 \times \alpha_u$ 则 25：更新阈值 $k_{attr} = k_{attr} + \alpha_u$26：否则 ▷不存在数据排除问题 27：退出循环 28：生成附加数据 $D_{aug} \leftarrow$ 运行 vae(D_{clean}, N_{aug}) 29：合并数据集 $D_{final} \leftarrow D_{dirty} \cup D_{aug}$

–

图5. 基于集成的错误检测算法

所有数据排除问题均已解决。随后，在执行典型的数据整理和特征工程任务之前，将最终的干净子集与脏数据合并。

IV. 干净数据增强

在本节中，我们将讨论干净数据增强模块的技术细节。在提取干净子集后，将使用数据增强模块生成与干净子集具有相同分布的数据。为此，我们研究了三种专用于表格数据的数据增强方法，包括 MODALS [6],变分自动编码器（VAE） [18]，和条件表格生成对抗网络（CTGAN） [12]。通过实验，我们发现 VAE 的性能优于另外两种方法[1]。具体而言，我们在一个

[1]为简洁起见，本文省略了三种增强方法之间比较研究的结果。

机器学习流水线中针对多个数据集研究了这三种方法。VAE 数据增强取得了比另外两种方法更高的预测准确率（F1 分数为 $\geq$ 0.98%）。因此，我们决定将 VAE 方法与自适应错误检测相结合。

通常，自动编码器由两个组件构成，即编码器和解码器。编码器将输入数据集压缩到低维空间中，该空间称为潜在空间表示。随后，解码器利用潜在空间表示恢复输入数据集。为此，自动编码器通常使用重构损失函数，将原始数据与解码器生成的值进行比较。在数据增强中，自动编码器会对潜在空间进行一些变换。为此，VAE模块训练其编码器，以提取数据分布的参数，而不是潜在空间表示[18]。如图2所示，解码器生成一个由均值 μ_x 和方差σ_x组成的二维向量。随后，VAE根据使用所提取参数形成的高斯分布 $\mathcal{N}(\mu_x, \sigma_x)$生成一组数据实例。然后，将这组数据实例作为解码器的输入，以生成额外的干净数据。

除了重构损失外，VAE模块还利用KL散度来区分原始数据集（即 AutoCure 中的干净子集）与生成数据的概率分布。通常，KL散度是一种统计量，用于量化两个概率分布与参考分布之间的差异。VAE模块通过优化均值和方差来模拟输入分布，从而努力降低KL散度的值。除优化指标外，从由生成的均值和方差参数化的分布中采样数据的过程是不可微的。在这种情况下，难以通过随机节点 Z 执行反向传播（这是优化编码器和解码器权重所必需的）。为解决这一问题，采用重参数化过程，使反向传播能够穿过随机节点。为此，重参数化器将随机节点 $Z \sim \mathcal{N}(\mu_x, \sigma_x)$ 转换为可微函数 $Z = \mu + \sigma \odot \epsilon$，其中 $\epsilon \sim \mathcal{N}(0, 1)$ 表示标准高斯分布；该分布与梯度计算无关（而梯度计算是反向传播过程中的必要步骤）。

V. 性能评估

在本节中，我们评估 AutoCure 相对于一组基线方法的有效性。通过这些实验，我们希望回答以下问题：（1）就其对下游机器学习模型的影响而言，AutoCure 与基线方法相比表现如何（此类模型的性能通过预测准确率和训练时间来量化）？（2）要达到与真实值相当的性能，所需的增强后的干净数据的最小数量是多少？（3）投票阈值 k对检测准确率有何影响？（4）数据集的错误率对 AutoCure 的性能有何影响？（5）AutoCure 如何提升模型性能？我们首先

介绍评估设置，然后讨论本研究过程中得到的结果和经验教训。

A. 实验设置

我们使用六个涵盖不同数据规模和不同错误率的真实数据集对 AutoCure 进行了评估。表 I 总结了每个数据集的特征。其中三个数据集，即 Adult、乳腺癌和智能工厂，与分类（CL）任务相关。而另外三个数据集，即 NASA、住房和土壤湿度，则包含回归任务（REG）。为控制评估环境，我们选择向数据集中注入不同的真实错误。为此，AutoCure 借助 BART 工具 [4] ，该工具能够系统地控制错误数量以及这些错误的修复难度。为了使用 BART 注入错误，我们采用一组否定约束来生成不同的错误类型，例如规则违反、离群值和缺失值。

表 I 评估中使用的数据集，其中 γ 表示错误率（%），错误类型包括规则违反（RV）、离群值（OT）、缺失值（MV）和拼写错误（TP）

数据集	Rows	列	错误类型	错误率	机器学习任务
Adult [19]	45223	15	RV、OT	0.09	CL
乳腺癌 [10]	700	12	MV、TP、OT	0.4	CL
智能工厂 [5]	23645	19	MV、OT	0.83	CL
NASA [34]	1504	6	MV、OT、TP	0.13	REG
住房 [14]	507	14	MV、OT	0.24	REG
土壤湿度 [31]	679	129	MV、OT	0.3	REG

我们将 AutoCure 的性能与一系列由不同错误检测与修复工具组成的基线方法进行了比较。表 II 列出了错误检测与修复方法。每种检测方法与修复方法的组合代表一种基线方法。例如，基线 B2 表示使用 dBoost 检测脏数据集中的错误，并使用基于机器学习的插补算法生成修复候选项。在修复方法列表中，我们还纳入了真实值和脏数据情况，以展示性能的上限和下限。标准插补器使用数值列的均值替换错误数据实例，并使用一个虚拟字符串替换分类列中的错误数据实例。作为替代方案，基于机器学习的插补器使用决策树模型预测错误的数值实例，同时使用 MissForest [33] 处理分类数据。评估所采用的指标包括检测准确率，即以检测精确率、召回率和 F1 分数表示的指标，以及训练时间和预测准确率。

对于VAE模块，编码器和解码器均已实现为前馈神经网络。VAE模块的输入包括训练集、测试集、干净子集的维度、隐藏层中的节点数以及潜在因子的数量。编码器输入层和解码器输出层中的节点数已设置为干净子集中的属性数量。编码器和解码器均包含两个隐藏层，第一个和第二个隐藏层中的节点数分别设置为50和12

表 II 用作基线的错误检测与修复工具

Idx	检测器	Idx	检测器	Idx	修复方法
B	dBoost [24]	M	Min-K [2]	-	脏数据
E	ED2 [26]	N	NADEEF [8]	1	真实值
F	FAHES [29]	Q	离群值：四分位距	2	机器学习插补器
H	HoloClean [30]	R	RAHA [23]	3	标准插补器
I	离群值：孤立森林 [36]	S	离群值：标准差	4	BARAN [22]
K	KATARA [7]	V	MV-Detector		

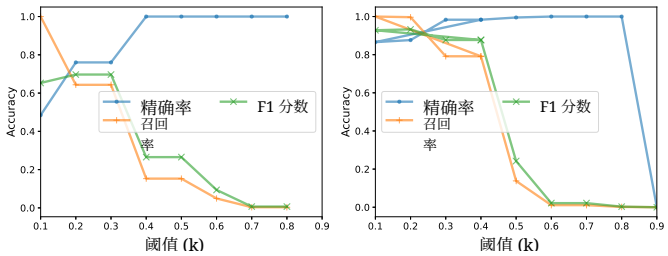
。我们采用Adam优化器来优化参数和自定义损失函数，该损失函数由均方误差和KL散度组合而成。我们在编码器和解码器的每个隐藏层中添加了ReLU激活函数。

为了考察 AutoCure 和基线方法的性能，我们使用 Keras 实现了一个神经网络，可处理回归、二分类和多分类任务[2]。AutoCure 利用一种称为 Optuna [3],的基于贝叶斯的知情搜索方法，来调整学习率、隐藏层数量以及每层的单元数。所有实验的训练轮次均固定为 500。每项实验均重复进行十次，并报告十次运行的平均值。为清晰起见，当标准差低于 1% 时，我们不予列出。所有实验均在一台运行 Ubuntu 16.04 LTS 的机器上进行，该机器配备 32 个 2.60 GHz 核心和 128 GB 内存。

B. 结果

a) 投票阈值： 在深入探讨 AutoCure 和基线方法的实验结果之前，我们首先评估改变投票阈值所产生的影响，以此为 AutoCure 中采用的自适应投票机制提供合理依据。图 6a 和图 6b 分别展示了传统 Min-k 检测方法在住房和 Smart Factory 数据集上检测错误实例时的检测准确率。这两幅图均表明，提高投票阈值 k 会对检测召回率（定义见第二节）和检测精确率（定义见第三节）产生不同的影响。例如，图 6a 显示，提高投票阈值 k 通常会降低检测召回率，因为被标记为错误的数据实例会减少。另一方面，提高阈值 k 会提升精确率，这是因为多种错误检测方法通过共识能够精确估计错误实例。鉴于这种权衡，在特定应用场景中准确选取最优 k 值往往并非易事。此外，我们在实验中注意到，像传统 Min-K 方法那样将 k值硬编码，通常会妨碍我们获得均衡的 干净数据比例。因此，AutoCure 会迭代调整 k 值，以应对相关的数据排除问题。

[2]AutoCure 的源代码、基线方法和数据集将在论文最终版本发布时公开。



（a）住房　　（b）智能工厂

图 6. 不同阈值 k （Min-k）下的检测准确率

b) 预测准确率：在这组实验中，我们评估了在所研究数据集的不同版本上训练的神经网络的准确率。对于 AutoCure，我们将所有数据集的生成实例数量设为 6000。图 7 展示了在干净数据集上训练的神经网络的预测性能[3]（参见图 7a 中的绿色区域），以及在一组已处理数据集上训练的神经网络的预测性能（参见图 7a 中的蓝色区域）。已处理数据集包括由 AutoCure 清理的数据集（在图 7 中简称为 A），以及使用基线工具的不同组合清理的数据集。对于 Adult 数据集，图 7a 根据神经网络的 F1 分数，对比了 AutoCure、干净数据集和 24 种基线方法的性能。如图所示，AutoCure 达到的准确率与干净数据集相近（两者的平均 F1 分数均为 89.5%）。值得注意的是，某些清理工具组合的性能与 AutoCure 相近，而其他组合（例如 Q2、Q3、K2 和 K3 ）则未能准确清理脏数据集。

图 7b 展示了使用乳腺癌数据集对 AutoCure 和基线方法进行的类似评估。同样，AutoCure 的表现优于所有基线方法，并且达到了与干净数据集相近的性能（AutoCure 的平均 F1 分数为 93.6%，而干净数据集为 94%）。图 7c 展示了使用智能工厂数据集对所比较方法进行的评估。显然，AutoCure 的表现优于大多数基线方法以及干净数据集（平均高出 46%）。需要强调的是，一些基线方法（例如 E2、H2 和 N2）的 F1 分数达到了 1。然而，这些基线方法的这种表现很大程度上取决于数据集。

对于 NASA 数据集，AutoCure 的表现优于大多数基线方法（AutoCure 的平均均方误差为 9.3，而干净数据集的平均均方误差为 15.3）。对于土壤湿度数据集，AutoCure 达到了合理的准确度（平均均方误差为 11.7），相较之下，干净版本的平均均方误差为 2.78，而部分基线方法的平均均方误差分别为 3.15 和 3.55（N2 和 H2）。事实上，需要强调的是，AutoCure 在不同数据集上取得的结果具有一致性。相比之下，其他基线方法缺乏这种一致性，例如，N2 和 H2 在 NASA 数据集上的表现较差（图 7d）。最后，图 7f 展示了

[3]在此语境下，干净数据集是指数据的真实值版本。

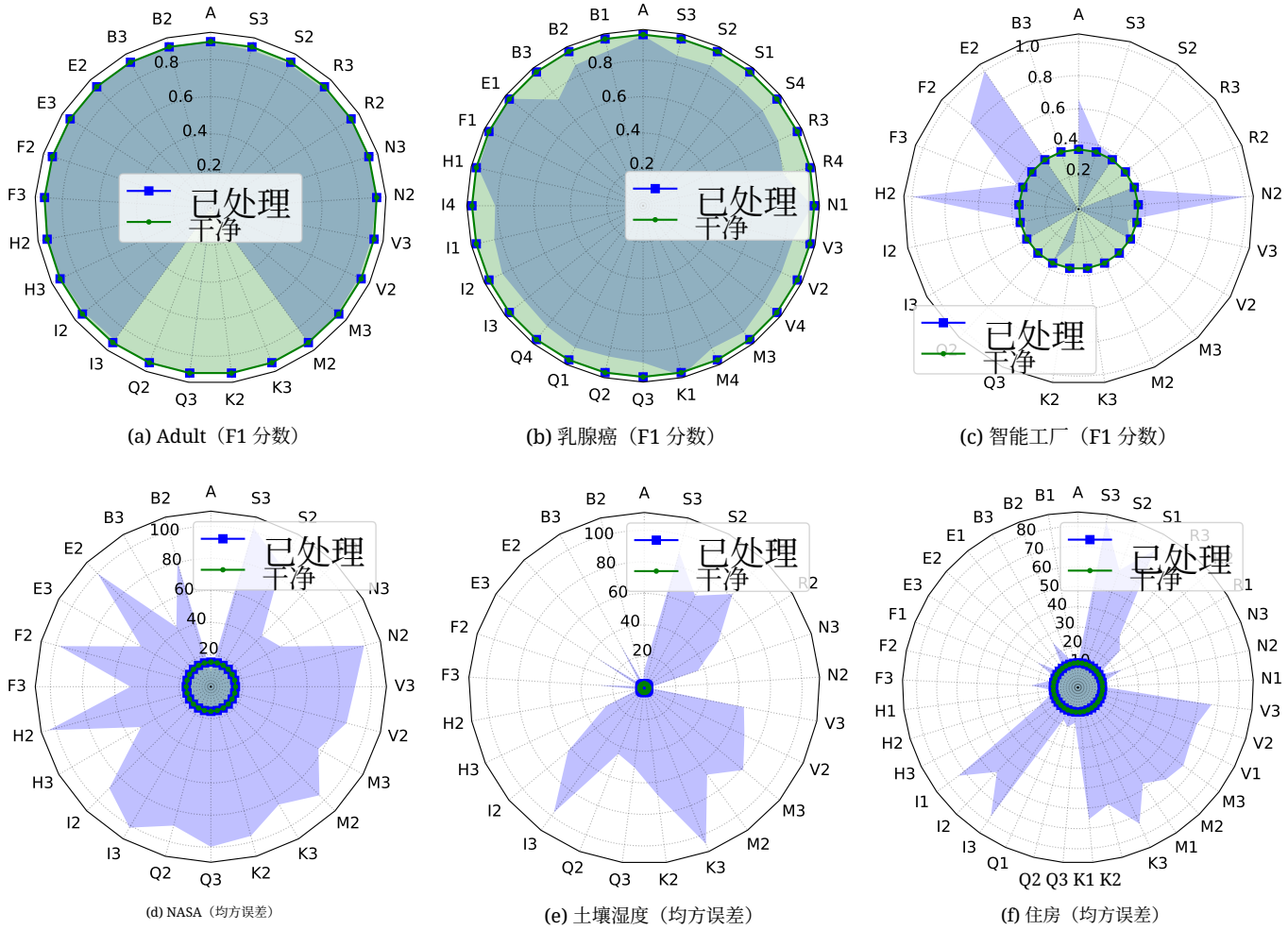
住房数据集上的均方误差结果。同样，AutoCure 的表现与干净数据集相似（两者的平均均方误差均为 12.2）。

c) 训练时间：除预测准确率外，我们还从训练时间的角度对 AutoCure 和基线方法进行了评估。表 III 列出了在干净数据集和经整理的数据集上训练的神经网络的平均训练时间。由于采用了数据增强，AutoCure 会导致所有数据集的训练时间相对增加。然而，对于某些数据集（如 Adult ），AutoCure 情况下的训练时间仅略高于使用干净数据集时的训练时间（平均约高出 2.9 分钟）。此外，公平的比较还应考虑一种现实场景：数据科学家通常并不了解应在其 ML 流水线中实施哪些数据整理工具。由于存在多种传统数据整理方法，因此有必要像 BoostClean [20] 和 ActiveClean[21] 一样，检查并执行多种传统数据整理方法。因此，对传统数据整理方法所需训练时间进行实际估算时，应综合计算所有训练时间。在这种情况下，对于所有数据集，AutoCure 的表现均显著优于传统数据整理方法（在表 III 中称为“组合”）。

表 III 比较方法的平均训练时间

	Adult	癌症	Nasa	湿度	工厂	住房
干净	1416.8	35.5	51.9	51.4	996	43.5
B2	1452.6	35.7	65.4	55.9	1265.8	45.65
B3	1467.2	41.4	63.2	52.6	1138.7	46.45
E2	1488.9	41.8	66.5	55.9	1138.2	47.08
E3	1487.6	37.1	54	51.6	1095.3	47.8
H2	1487.4	42.65	65.77	55.3	1152.4	45.83
H3	1495.8	33	53.89	55.4	1023.6	46.74
K2	1423.4	43.9	66.1	50.7	1105.7	42.85
K3	1370.5	38.1	63	46.3	1097.1	40.24
R2	1488.6	43.9	64.8	56.3	946.1	47.21
R3	1439.6	41.1	64.8	53.1	1166.9	45.61
组合	14601	398.4	627.5	533.1	11129.8	455.46
AutoCure	**1595.2**	**233.7**	**183.1**	**300.9**	**1352**	**300.6**

d) 增强规模： 在这组实验中，我们针对每个数据集评估增强后的干净数据规模对预测性能和训练时间的影响。实验得出的一般结论是，增加增强数据量通常能够提升预测性能，但代价是训练时间延长。例如，图 8a 展示了在 Adult 数据集的干净数据版本上训练的神经网络的 F1 分数（参见红色虚线），以及 AutoCure 使用不同规模的增强后的干净数据整理出的多个版本的 F1 分数。图中显示，只需少量增强数据（约为 Adult 数据集规模的 1%），即可达到与干净数据集相同的性能。在 Smart Factory 数据集上也获得了类似结果：只需达到干净数据集规模 1% 的增强数据，即可达到干净数据集的质量。



(a) Adult（F1 分数）　(b) 乳腺癌（F1 分数）　(c) 智能工厂（F1 分数）

(d) NASA（均方误差）　(e) 土壤湿度（均方误差）　(f) 住房（均方误差）

图 7. 干净数据集的建模准确率、AutoCure（记为“A”）以及一组基线方法

对于规模较小的数据集，实验显示需要生成更多干净数据实例。在乳腺癌数据集的情况下，图 8c 表明，AutoCure 需要更多干净实例（至少达到原始数据集规模的 375%），才能取得与干净数据集相近的准确率。对于 NASA 数据集，当生成的干净数据规模足够大（约为原始数据集规模的 200%）以减小错误实例的影响时，AutoCure 能够达到与干净数据集相当的性能。在土壤湿度数据集上也获得了类似结果（参见图 8e）。

图 9 展示了 AutoCure 策划的不同数据集的训练时间。例如，图 9a 展示了干净版本（参见红色虚线）以及 AutoCure 策划的多个版本的训练时间。正如图中所示，AutoCure 所需的时间比该数据集的干净版本多约三分钟。对于 Smart Factory 数据集，为满足质量要求，AutoCure 所需的时间略长（最多多 18.4%）。对于乳腺癌数据集，为了达到与干净数据相近的准确率（参见图 9c），AutoCure 需要额外花费 100 秒。同样，对于 NASA 数据集，AutoCure 还需要额外花费 80 秒（参见图 9d）。对于土壤湿度数据集，由于其关联的回归任务，AutoCure 必须生成大量干净实例。因此，AutoCure 需要额外花费约 400 秒的训练时间，才能达到性能上限。

e) 错误率：在这组实验中，我们评估了错误率对 AutoCure 及一组基线方法性能的影响。我们固定生成的干净实例数量，并逐步增加注入错误的数量（即缺失值和数值异常值）。图 10 展示了神经网络的预测准确率：对于回归任务，采用均方误差（MSE）表示；对于分类任务，采用 F1 分数表示。神经网络分别基于干净数据集，以及使用 AutoCure 和基线方法整理后的数据集进行训练。在图 10a 中，AutoCure 表明其性能基本不受 NASA 数据集中的错误数量影响。即使生成的干净实例数量保持不变，AutoCure 的曲线也仅略有上升，且仍低于代表干净数据集的红色虚线。相反，随着错误率提高，所有传统数据整理方法的表现都较差。

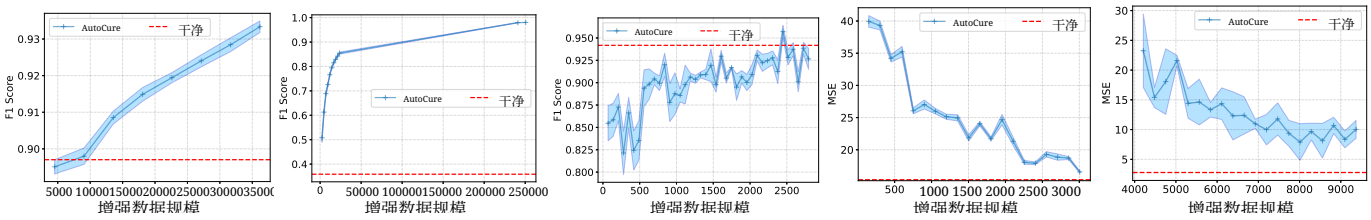


图 8. 不同增强后的干净数据规模下 AutoCure 的建模准确率

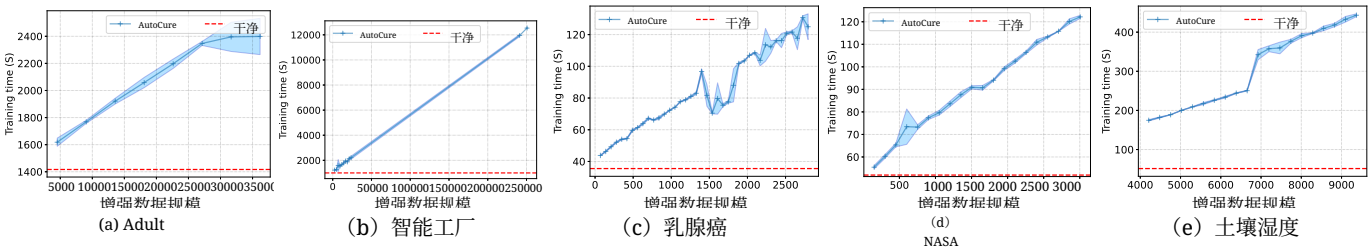


图 9. AutoCure 在不同规模的增强干净数据上的训练时间

图 10b 和图 10c）显示了住房和乳腺癌数据集的类似结果。图 11a 至图 11c 展示了错误率提高过程中神经网络的训练时间。显然，提高错误率几乎不会影响所有受测数据集的训练时间（某些错误率下偶尔出现的尖峰除外）。

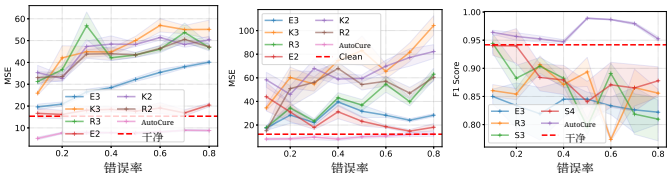


图 10. 错误率对预测准确率的影响

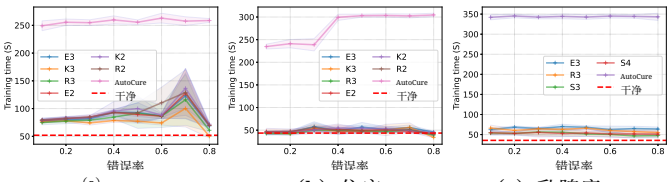


图 11。错误率对训练时间的影响

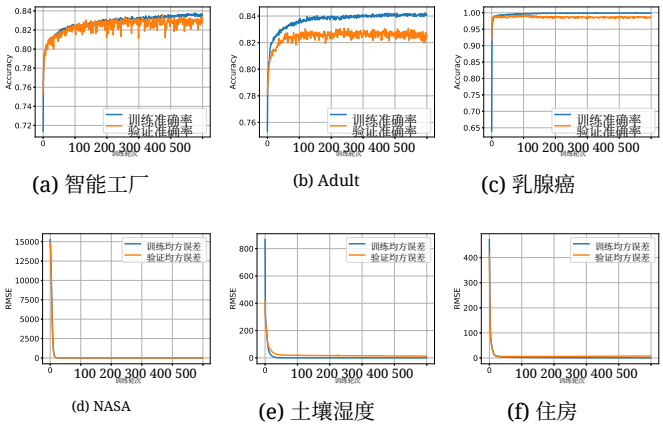
f) 过拟合：如上所述，结果表明，AutoCure 无需任何修复操作，便能全面提升预测性能。然而，如此高的性能可能是过拟合造成的。因此，在这组实验中，我们考察了使用 AutoCure 生成的神经网络模型是否存在过拟合。图 12 展示了学习曲线随训练轮次以及预测性能的变化，涉及多个数据集。例如，对于 Smart Factory 数据集，图 12a 显示，模型的二元准确率随时间不断提高，直至趋于平稳。显然，训练集和验证集的两条曲线彼此接近。其他所有数据集也都出现了相同的结果（参见图 12b-12f）。因此，图 7 所示 AutoCure 的高性能并非过拟合所致，主要得益于干净数据实例的高密度。

C. 结果总结

本节重点介绍本研究获得的主要发现。第一个发现围绕性能一致性展开。具体而言，我们的结果表明，一些基线方法可能达到与 AutoCure 相近甚至更好的性能（例如，图 7c 中使用 E2、E3 和 H2 整理后的数据训练的模型的准确率）。不过，需要指出的是，这些结果在很大程度上取决于数据集，即性能可能因数据集而异。此外，还必须进行多项实验，以确定适用于每个数据集的整理策略。另一方面，结果表明，AutoCure 在不同数据集上表现出结果的一致性。对于所有经过检验的数据集，AutoCure 都通过突出干净数据比例，成功提升了数据集的质量。

另一个重要发现是，小型数据集通常需要生成更多干净数据实例（参见图 8c）。AutoCure 出现这种行为的原因是，在小型数据集中，原始干净实例的比例通常较低。因此，AutoCure 会生成大量干净实例，以满足质量要求。该发现同样适用于具有关联回归任务的数据集，例如 Housing、Soil Moisture 和 NASA（参见图 8d 和图 8e）。在

事实上，正如我们的实验所发现的，与分类任务相比，回归模型通常对数据错误更加敏感。对于大型数据集，例如 Adult 和 Smart Factory，AutoCure 只需生成少量数据，即可达到与干净数据集相近的准确率。此外，结果还表明，在某些情况下，由于训练实例数量增加，AutoCure 会延长训练时间（参见图 9c、图 9d 和图 9e）。最后，结果证实，即使错误率较高，AutoCure 仍能表现良好，如图 10 所示。



(a) 智能工厂 (b) Adult (c) 乳腺癌

(d) NASA (e) 土壤湿度 (f) 住房

图 12. 使用 AutoCure 训练的神经网络的学习曲线

VI. 相关工作

本节介绍与数据整理问题相关的先进技术和工具。事实上，学术界和工业界存在大量数据整理技术和工具。例如，HoloClean [30]是一种与机器学习无关的数据修复技术，通过综合利用多个清理信号来构建概率图模型，从而推断修复值。为修复模式违规和不一致性，OpenRefine [13]使用 Google Refine 表达式语言（GREL）作为其原生语言，以转换现有数据或创建修复值。类似地，BARAN [22] 是一种用于修复不同错误类型的整体式、无需配置的基于机器学习的方法。为此，BARAN 增量式地训练可更新的模型，利用数据错误的值、邻近信息和领域上下文来提出纠正候选项。为进一步增加训练数据，BARAN 利用外部来源，例如维基百科页面修订历史。

事实上，上述介绍的技术，如 HoloClean、OpenRefine 和 BARAN，并未考虑下游机器学习应用提出的要求。无论数据来自何处、以何种方式产生，或数据将如何被使用，它们往往都会改进数据质量。因此，一组新的技术和工具应运而生，致力于联合优化清洗任务和建模任务。换言之，这些面向机器学习的方法专注于选择最优的修复候选项，以提升特定预测模型的性能。相应地，这些方法假设可以获得修复候选项这些候选项来自其他与机器学习无关的方法。例如，BoostClean 将错误修复任务视为统计提升问题。具体而言，它将一组弱学习器组合成一个强学习器。为了生成弱学习器，BoostClean 迭代选择一对检测方法和修复方法，然后将其应用于训练集，以推导出一个新模型。

ActiveClean [21]是另一种面向机器学习的方法，主要用于采用凸损失函数的模型。它将数据清洗任务建模为一个随机梯度下降问题。首先，它在脏训练集上训练一个模型，并迭代更新该模型，直至达到全局最小值。在每次迭代中，ActiveClean 会抽取一组记录，然后请求预言机对其进行清洗，以沿最陡梯度方向更新模型。类似的工作 CPClean [17] 会逐步清洗训练集，直到能够确定进一步修复不会改变模型预测。事实上，面向机器学习的方法并未引入新的数据修复技术，例如 HoloClean 和 BARAN。相反，这些方法往往会从已有的修复候选项中进行选择，以提升预测性能。与 AutoCure 相比，我们的解决方案无需生成修复候选项，因此可以完全避免使用这类方法。因此，AutoCure 无需面对搜索修复候选项以及从中选出最佳子集所带来的复杂性。此外，面向机器学习的方法通常针对特定的优化方法和机器学习模型量身定制，例如 ActiveClean 仅适用于凸损失函数问题。相比之下，AutoCure 可用于所有机器学习任务和优化方法。

VII. 结论与未来工作

本文介绍了一种处理错误数据的新方法，称为 AutoCure。作为传统数据整理工具的替代方案，AutoCure 无需识别检测到的错误类型，因为它会完全忽略这类含噪数据实例。相反，它处理干净实例；如果确实存在更多干净实例，这将有助于提升建模性能。此外，AutoCure 避免了在庞大的修复候选项空间中进行搜索以及选择最合适候选项的复杂过程。作为概念验证，我们使用六个数据集以及 28 种以上的数据整理工具组合对 AutoCure 进行了评估。结果表明，使用 AutoCure 整理后的数据集能够达到与干净数据集相当甚至更好的性能。尽管 AutoCure 在多个数据集上表现出色，但仍可通过以下方式进一步改进：(1) 将其与数据估值工具集成，以减少训练时间；(2) 根据脏数据集的规模及相关机器学习任务，动态调整需要增强的实例数量。

致谢

本研究得到联邦教育与研究部的资助（部分资助），资助编号为 02L19C155、01IS21021A（ITEA 项目编号 20219）。

参考文献

[1]Mohamed Abdelaal、Christian Hammacher 和 Harald Schoening。《REIN：机器学习流水线数据清洗方法的综合基准框架》。载于 第*26*届扩展数据库技术国际会议（*EDBT*），2023年3月。[2] Ziawasch Abedjan、Xu Chu、Dong Deng、Raul Castro Fernandez、Ihab F Ilyas、Mourad Ouzzani、Paolo Papotti、Michael Stonebraker 和 Nan Tang。《检测数据错误：我们目前处于什么阶段，还需要做什么？》《*VLDB*基金会会刊》，9(12):993–1004，2016年。[3] Takuya Akiba、Shotaro Sano、Toshihiko Yanase、Takeru Ohta 和 Masanori Koyama。《Optuna：新一代超参数优化框架》。载于 第*25*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集，2019年。[4] Patricia C Arocena、Boris Glavic、Giansalvatore Mecca、Ren´ee J Miller、Paolo Papotti 和 Donatello Santoro。《扰乱BART：用于评估数据清洗算法的错误生成》。《*VLDB*基金会会刊》，9(2):36–47，2015年。[5] Oliver Birgelen、Alexander；Niggemann。《智能工厂：用于能源优化的高存储系统数据》，2018年。访问于2022年1月。[6] Tsz-Him Cheung 和 Dit-Yan Yeung。《MODALS：潜在空间中与模态无关的自动数据增强》。载于 学习表征国际会议，2020年。[7] Xu Chu、John Morcos、Ihab F Ilyas、Mourad Ouzzani、Paolo Papotti、Nan Tang 和 Yin Ye。《KATARA：由知识库和众包驱动的数据清洗系统》。载于 *2015*年*ACM SIGMOD*数据管理国际会议论文集，第1247–1261页，2015年。[8] Michele Dallachiesa、Amr Ebaid、Ahmed Eldawy、Ahmed Elmagarmid、Ihab F Ilyas、Mourad Ouzzani 和 Nan Tang。《NADEEF：商品化数据清洗系统》。载于 *2013*年*ACM SIGMOD*数据管理国际会议论文集，第541–552页，2013年。[9] Shi Dong、Ping Wang 和 Khushnood Abbas。《深度学习及其应用综述》。《计算机科学评论》，40:100379，2021年。[10] Dheeru Dua 和 Casey Graff。《UCI机器学习资料库》，2017年。[11] Daniel Del Gaudio、Tim Schubert 和 Mohamed Abdelaal。《RTClean：使用实时OFD的上下文感知表格数据清洗》。载于 *2023*年*IEEE*普适计算与通信国际会议研讨会及其他附属活动（*PerCom*研讨会）。IEEE，2023年3月。[12] Ian Goodfellow、Jean Pouget-Abadie、Mehdi Mirza、Bing Xu、David Warde-Farley、Sherjil Ozair、Aaron Courville 和 Yoshua Bengio。《生成对抗网络》。《*ACM*通讯》，63(11):139–144，2020年。[13] Kelli Ham。《OpenRefine（2.5版）》。http://openrefine.org。用于清洗和转换数据的免费开源工具。《医学图书馆协会杂志：*JMLA*》，101(3):233，2013年。[14] D. Harrison 和 D.L. Rubinfeld。《波士顿住房数据》，1993年。访问于2021年2月。[15] Alireza Heidari、Joshua McGrath、Ihab F Ilyas 和 Theodoros Rekatsi- nas。《HoloDetect：用于错误检测的小样本学习》。载于 *2019*年国际数据管理会议论文集，第829–846页，2019年。[16] Ihab F Ilyas 和 Felix Naumann。《数据错误：症状、原因与起源》。《数据工程》，第4页，2022年。[17] Bojan Karla˘s、Peng Li、Renzhi Wu、Nezihe Merve G¨urel、Xu Chu 和 Ce Zhang。《不完整信息上的最近邻分类器：从确定答案到确定预测》。*arXiv*预印本 *arXiv:2005.05117*，2020年。[18] Diederik P Kingma 和 Max Welling。自编码变分贝叶斯。*arXiv* 预印本 *arXiv:1312.6114*，2013年。[19] Ron Kohavi 等。提升朴素贝叶斯分类器的准确率：决策树混合模型。载于 *KDD*，第 96 卷，第 202–207 页，1996 年。[20] Sanjay Krishnan、Michael J Franklin、Ken Goldberg 和 Eugene Wu。BoostClean：面向机器学习的自动错误检测与修复。*arXiv* 预印本 *arXiv:1711.01299*，2017 年。[21] Sanjay Krishnan、Jiannan Wang、Eugene Wu、Michael J Franklin 和 Ken Goldberg。ActiveClean：用于统计建模的交互式数据清洗。*VLDB*基金会会刊，9(12)：948–959，2016 年。[22] Mohammad Mahdavi 和 Ziawasch Abedjan。BARAN：通过统一的上下文表示和迁移学习实现有效的错误纠正。*VLDB*基金会会刊，13(12)：1948–1961，2020年。[23] Mohammad Mahdavi、Ziawasch Abedjan、Raul Castro Fernandez、Samuel Madden、Mourad Ouzzani、Michael Stonebraker 和 Nan Tang。RAHA：无配置错误检测系统。载于 *2019* 年 第 882 届，2019 年，第 865国际数据管理会议页 – 。[24] Zelda Mariet、Rachael Harding、Sam Madden 等。使用自动元组扩展检测异构数据集中的异常值，2016 年。[25] Felix Neutatz、Binger Chen、Ziawasch Abedjan 和 Eugene Wu。从机器学习前的数据清洗到面向机器学习的数据清洗。*IEEE* 数据工程公告，44(1)：24–41，2021 年。[26] Felix Neutatz、Mohammad Mahdavi 和 Ziawasch Abedjan。ED2：错误检测中的主动学习案例。载于 第 *28* 届 *ACM* 国际信息与知识管理会议论文集，第 2249–2252 页，2019 年。[27] Thorsten Papenbrock、Tanja Bergmann、Moritz Finke、Jakob Zwiener 和 Felix Naumann。使用 Metanome 进行数据剖析。*VLDB*基金会会刊，8(12)：1860–1863，2015年。[28] Eduardo HM Pena、Eduardo C de Almeida 和 Felix Naumann。近似（及精确）否定约束的发现。*VLDB*基金会会刊，13(3)：266–278，2019年。[29] Abdulhakim A Qahtan、Ahmed Elmagarmid、Raul Castro Fernandez、Mourad Ouzzani 和 Nan Tang。FAHES：一种稳健的伪装缺失值检测器。载于 第 *24* 届 *ACM SIGKDD* 国际知识发现与数据挖掘会议论文集，第 2100 2109 页，2018 年 – 。[30] Theodoros Rekatsinas、Xu Chu、Ihab F Ilyas 和 Christopher R´e。HoloClean：通过概率推断实现整体数据修复。*arXiv* 预印本 *arXiv:1702.00820*，2017 年。[31] Felix M. Riese 和 Sina Keller。利用高光谱数据进行土壤湿度回归的自组织映射框架。载于 *IGARSS 2018 - 2018 IEEE* 国际地球科学与遥感研讨会，第 6151–6154 页，西班牙瓦伦西亚，2018 年 7 月。[32] Claude Elwood Shannon。通信的数学理论。*ACM SIGMOBILE* 移动计算与通信评论，5(1)：3–55，2001 年。[33] Daniel J Stekhoven 和 Peter B¨uhlmann。MissForest——混合类型数据的非参数缺失值插补。生物信息学，28(1)：112–118，2012 年。[34] D. Stuart Pope、Thomas F. Brooks 和 Michael A. Marcolini。NASA翼型自噪声数据集，2014 年。访问于2022年1月。[35] De Jong Yeong、Gustavo Velasco-Hernandez、John Barry 和 Joseph Walsh。自动驾驶车辆中的传感器与传感器融合技术：综述。*Sensors*，21(6)：2140，2021年。[36] Ji Zhang。异常值检测的进展：综述。可扩展信息系统 *ICST* 汇刊，13(1)：1–26，2013年。