《实证软件工程》稿件编号（由编辑填写）

机器学习应用中的日志实践研究

Patrick Loic Foalem · Foutse Khomh · Heng Li

收稿日期：日期 / 接受日期：日期

arXiv:2301.04234v1 [cs.SE] 10 Jan 2023

摘要 日志记录是传统软件开发中的常见实践。已有多项研究探讨传统软件系统中的日志记录实践的不同特征（例如 Android 应用、JAVA 应用、C/C++ 应用）。如今，基于机器学习的应用（机器学习应用）正不断涌现。当前，许多流行的库为此类应用的开发提供了便利并作出贡献，其中包括：PyTorch、TensorFlow、Theano、MXNet、Scikit-Learn、Caffe 和 Keras。尽管机器学习广受欢迎，我们对机器学习应用中的日志记录实践仍缺乏清晰认识。本文旨在填补这一知识空白，帮助机器学习从业者了解基于机器学习的应用中日志记录的特征。具体而言，我们对 110 个开源基于机器学习的应用程序开展了实证研究。通过定量分析，我们发现，与 Android、JAVA 和 C/C++应用等传统应用相比，基于机器学习的应用中的日志记录实践并不那么普遍。此外，与传统应用相比，基于机器学习的应用中的大多数日志语句处于信息 和警告 级别；而在传统应用中，信息 是 C/C++ 应用中占比最高的日志语句级别，调试和错误 级别则构成了 Android 应用中占比最高的日志语句级别。我们还对随机抽取的日志语句样本进行了定量和定性分析，以了解机器学习开发者在哪里放置了大多数日志语句，并探究他们为何以及如何使用日志记录。这些分析得出了以下观察结果：(i) 机器学习开发者将大多数日志语句放在模型训练和非机器学习组件中。(ii) 数据和模型管理

*通讯作者。

Patrick Loic Foalem、Foutse Khomh、Heng Li 蒙特利尔理工学院计算机工程与软件工程系 加拿大魁北克省蒙特利尔 电子邮件：{patrick-loic.foalem, foutse.khomh,heng.li}@polymtl.ca

似乎是基于机器学习的应用中引入日志语句的主要原因。事实上，机器学习开发者使用日志语句跟踪数据处理的不同阶段，并记录相关统计信息。此外，他们使用日志语句跟踪所有已执行的实验，以找到最佳模型，并记录模型在验证数据或测试数据上的所有性能表现。我们希望通过这项工作增进对基于机器学习的应用中日志记录实践的了解。我们的研究结果还凸显了开发更多机器学习专用日志库的必要性，以记录并自动化基于机器学习的应用流水线中执行的所有实验。

关键词 日志记录实践 · 基于机器学习的应用 · 软件仓库挖掘 · 源代码分析。

1 引言

在软件程序中插入日志语句是良好编程实践的一部分（Kernighan 和 Pike（1999年））。日志语句可以收集程序运行期间的行为信息。这些信息常用于软件维护和运维中的各种任务，例如诊断故障、报告错误、检测异常，以及利用日志数据自动生成日志文本（Yang 和 Agrawal，2016年；Nagaraj 等人，2012年；Xu 等人，2009年；Ding 等人，2022年）。许多研究都对传统软件开发中的日志记录实践进行了描述。例如，Yuan 等人（2012年）研究了以 C/C++ 编写的开源应用中的日志记录实践；Zeng 等人（2019年）对 Android 应用进行了类似研究；Chen 等人（2017年）则研究了基于 JAVA 的应用中的日志记录实践。然而，据我们所知，此前尚无研究考察基于机器学习的应用中的日志记录实践。本文旨在填补这一研究空白。此类研究有助于机器学习从业者了解机器学习应用中的日志记录实践、识别被记录的信息类型以及所使用的日志库。日志由日志语句生成。清单 1 展示了机器学习应用中使用的一些典型日志语句示例。第 1、3 和 9 行的日志语句使用 Python 编程语言的通用日志库，而第 5、7、11、13 和 15 行的日志语句使用机器学习专用日志库。通用日志库由 Python 模块 `logging` 构建。其日志语句始终包含日志级别（`INFO`、 `Warn`、 `ERROR`、`DEBUG`、 `CRITICAL`），而且通常早于机器学习专用日志库。这些库主要面向传统软件系统设计，而机器学习专用日志库则面向机器学习应用，包含机器学习专属信息，并提供了一种便捷的方法来记录性能、超参数以及有关数据的统计信息。一些机器学习日志库甚至还可以将记录的内容以图表形式可视化，让用户浏览模型的学习过程。

```
logging.info("*** Reading from input files ***")

tf.logging.info("[*] num_epochs: %d" % _num_epochs)

dllogger.log(step=(epoch, steps_per_epoch, batch_idx), data=dllogger_data)

caffe.log('Using devices %s' % str(gpus))

rospy.logwarn("Updating Steering PID %s, %s, %s", config["Steer_P"],
↪  config["Steer_I"], config["Steer_D"])

self.hparams.train_logger.log_stats({"Epoch": epoch, "lr": old_lr},
↪  train_stats={"loss": self.train_loss}, valid_stats=stats, )

mlflow.log_metrics(metrics, step=step)

wandb.log(log_data, step=learner.num_samples_collected + args.start_env_steps)
```

清单 1：从基于机器学习的开源应用中提取的日志语句示例。

为了解基于机器学习的应用中的日志记录实践，本文对来自 GitHub 的 110 个基于机器学习的软件项目开展了一项实证研究[1]；考察日志语句的普及程度、使用原因，以及发生日志记录的机器学习组件的特征。我们的研究围绕以下三个研究问题（RQ）展开：

研究问题1： 基于机器学习的软件日志实践具有哪些特征？为回答这一研究问题，我们统计了所研究的基于机器学习的系统中包含的日志语句，并将结果与以往关于传统系统（即 JAVA、C/C++、Android）的研究发现进行比较。我们发现，基于机器学习的应用中的日志密度与传统 Android、JAVA 和 C/C++ 应用的报告密度存在差异。具体而言，基于机器学习的应用中的日志记录不如 JAVA、Android 和 C/C++应用普遍。此外，基于机器学习的应用中的日志级别分布也不同于传统应用。

RQ2： 机器学习流水线的哪些阶段更容易进行日志记录？机器学习流水线包含以下步骤：数据收集、数据处理、模型训练、模型评估和模型部署。我们手动分析了 ∼2K 条日志语句，其置信水平超过 95%，置信区间为 5%（Yamane，1967年）。我们将每条日志语句映射到机器学习流水线的一个阶段，并检查了其内容。我们发现，大多数日志语句出现在模型训练步骤中，包含有关模型超参数和性能的信息。

[1] https://github.com/

RQ3： 为什么开发者会在基于机器学习的应用中使用日志记录？
为了理解使用日志语句背后的开发决策依据，我们检查了样本中包含 380 条日志语句的文件代码，并为每条日志语句的使用指定了原因。结果表明，机器学习从业者使用日志语句主要出于两个原因：数据管理和模型管理。

这些结果表明，基于机器学习的应用与广受研究的传统应用存在截然不同的日志记录实践，也说明有必要开展更多研究，以加深我们对基于机器学习的应用中日志记录实践的理解。机器学习代码中与通用日志库（用于代码和模型管理）相关的日志语句占比很高，这可能表明机器学习从业者并不了解机器学习专用日志库，例如 `mlflow`，`wandb`。例如，在列表 2中，机器学习从业者仍然使用通用日志库来记录模型在训练阶段的性能（即第 1 行），而没有使用专门用于记录此类信息的机器学习专用日志库，尽管这些专用库具有诸多优势；例如，可以一次记录多个机器学习指标、跟踪学习阶段执行的所有实验，并通过可视化对实验结果进行总结，以便跟踪性能改进（即第 3、5、6 和 8 行）。

```
1 logger.info('Epoch {} Loss {:.4f} Accuracy {:.4f}'.format( epoch,
2 loss_meter.avg, accuracy))
3 dllogger.log(step=(epoch, steps per epoch, batch idx),data=dllogger data,
4  verbosity=0)wandb.log({"RMSE_training": training_RMSE, "epoch": epoch,})
5 self.hparams.train_logger.log_stats({"Epoch": epoch, "lr": old_lr},
6 train_stats={"loss": self.train_loss}, valid_stats=stats, )
7 mlflow.log_metrics(metrics, step=step)
8
```

清单2：日志记录用于对机器学习性能进行日志记录的语句示例。。

论文结构。本文其余部分安排如下。第2 节介绍我们的研究设计。第3 节阐述研究动机以及各项研究问题的结果。第4 节讨论研究结果有效性面临的威胁。第5 节讨论相关文献，第6节总结全文。

2 研究设计

本节介绍了我们的研究设计，旨在了解基于机器学习的应用中的日志记录实践。

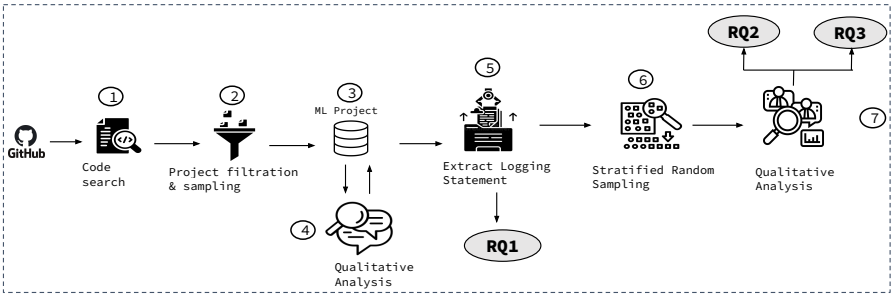


图 1：研究工作流程概览。

2.1 数据收集

图 1 概述了我们的数据提取与分析方法。下面将逐步进行说明。

① **代码搜索**
为了筛选用于本研究的基于机器学习的软件项目，我们使用了 GitHub API。GitHub 搜索 API V3[2] 允许我们通过代码搜索来收集项目。我们根据导入语句搜索项目。为获取基于机器学习的项目，我们采用与 Openja 等人（2022年）相同的方法，即在 GitHub 项目的导入语句中搜索机器学习库。具体而言，我们搜索对应以下机器学习库的导入语句：TensorFlow、Theano、MXNet、Scikit-Learn、PyTorch、Keras、Caffe。我们总共收集了 1,547 个基于机器学习的项目。选择这些库是因为它们广受欢迎。

② **项目筛选与抽样**。
为了在研究中保留数量适当的成熟项目，我们采用了以往研究（Openja 等人，2022年；Munaiah 等人，2017年）提出的筛选方法，即为以下三项指标设定阈值：星标数、提交数和作者数。为确定合适的阈值，我们绘制了这三项指标的分布图。图 2展示了我们从 GitHub 收集的所有项目按提交数排列的累积频率曲线。对于这项指标（即提交数），我们将阈值设为 100，以排除教程、学生课程作业等“玩具”项目。采用这一阈值后，我们保留了 320 个项目。我们还绘制了另外两项指标的累积频率曲线（相应图表见附录图 8），并将星标数和作者数的阈值分别设为 1 和 2。随后，我们从数据集中筛除了星标少于 1 个且作者少于 2 人的所有项目，最终保留了 110个项目。

③ **机器学习项目收集**

[2] https://docs.github.com/en/rest

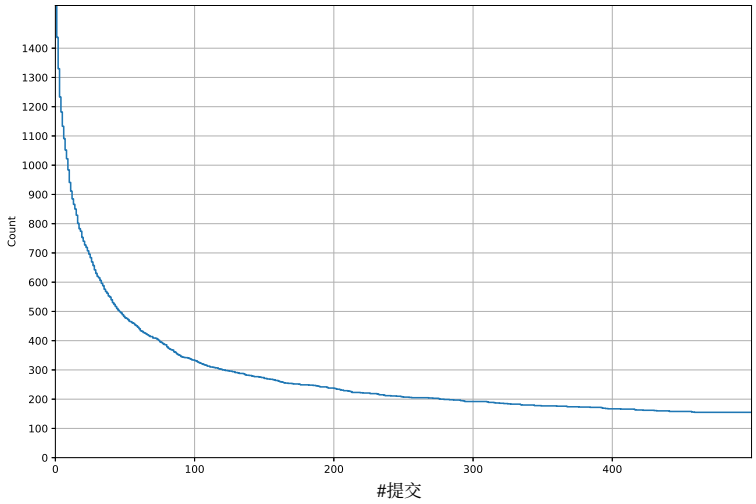


图 2：基于 #提交 的累积频率曲线

共有 110个项目符合我们的筛选标准，并被保留为本研究的研究对象。为确保所研究的系统中不存在分叉项目，并且我们的数据集中涵盖了步骤 1 中提及的至少一个机器学习库，本文第一作者手动检查了这 110个项目各自的代码仓库，所有 110 个研究项目均符合这些标准。

表 1 概述了我们选定的项目。行 **源代码行数** 给出了源代码行数的汇总统计量，包括研究对象的平均代码行数和最多代码行数。行 **#提交** 是提交次数的汇总统计量。行 **#Python 文件**表示分析时各研究对象中 Python 文件数量的汇总统计量。每个项目的贡献者数量汇总统计量列于行 **作者数量**。星标数量是衡量项目受欢迎程度的指标，行 **#星标** 是对研究对象星标数量的统计描述，平均星标数为 1,167。

④ **识别项目中包含的日志库**
为了获取每条日志语句的信息，我们使用 Python 内置包 AST 构建了一个自定义的抽象语法树（AST）解析器[3]，并用它提取所选项目中所有 Python 文件包含的导入语句。我们总共提取了 13,270 条导入语句（步骤 3 - 4，图 1 ）。在步骤 4 中，我们人工分析了

[3] https://bit.ly/3gxsOkc

表 1：研究对象概览

指标	Mean	Min	第 25 百分位数	中位数	第 75 百分位数	Max
SLOC	53,579	244	3,647	10,662	31,228	2,010,902
# 提交	759	42	172	323	782	9,139
# Python 文件	272	5	35	101	210	7,510
# 作者	18	2	3	5	13	261
# 星标	1,167	1	2	20	284	32,770

提取出的导入语句，以确定涉及的日志库。我们在数据集中的项目里识别出以下 12 个库：logging[4],rospy[5], 超参数[6],日志记录器[7],警告[8],回调[9], WANDB[10],TensorFlow[11],caffe.log[12], dllogger[13], ml-logger[14], MLflow。

⑤ **日志语句提取**

为统计每个文件中包含的日志语句，我们采取了以下步骤：(i) 对于这To identify 110个项目中的每个 Python 文件，我们使用 Python 3 的 AST模块为该文件生成 AST。(ii) 我们使用 Python抽象语法树库提取与上述 13 个日志库相关的函数调用。(iii) 我们分析了上一步中提到的每个库，然后确定了能够匹配这些库中日志语句的正则表达式。(iv) 因此，我们使用以下表达式：log、info、debug、error、fatal、warn、callbacks、warnings、exception 和 critical 作为访问者模式[16]，以获取所有与日志语句对应的函数调用。为尽量减少日志语句数据集中的误报，我们手动移除了包含“log”但并非日志语句的表达式（例如“dialog”“login”“numpy.log”“tensor.log”等），还移除了与日志配置对应的表达式（例如“caffe.init log()”“logging.getLogger”“logging.disable”“logging.config”等），这使我们得以保留一个干净的数据集[17]。这一日志语句提取过程借鉴了（Chen 等人，

[4] https://bit.ly/3gxsOkc
[5] https://bit.ly/3shluMk
[6] https://bit.ly/3sdKpk2
[7] https://bit.ly/3VMYr9K
[8] https://bit.ly/3CVpxCR
[9] https://bit.ly/3SozpLk
[10] https://bit.ly/3Sod8Of
[11] https://bit.ly/3guvYoR
[12] https://bit.ly/2QjqYSf
[13] https://bit.ly/3TJyW7m
[14] https://bit.ly/3zOXz7O
[15] https://bit.ly/2KiwDFd
[16] https://bit.ly/3TKBeD8
[17] https://bit.ly/3DlmGow

2017年；Zhai 等人，2019年）。我们总共获得了 ~21K 条日志语句。步骤5 的结果回答了我们的研究问题**研究问题1**。

⑥ **分层随机抽样**
为了回答研究问题 **RQ2** 和 **RQ3**，我们采用分层随机抽样法为定性分析准备数据集。我们选择分层抽样方法，以确保样本中的项目按比例得到代表。我们总共抽取了 ~2K 条日志语句，其置信水平超过 95%，置信区间为 5%（Yamane，1967年）。

2.2 数据分析

我们对步骤 6 中抽取的日志语句进行人工分析，将日志语句分类并映射到机器学习流水线的不同阶段（图 1 中的步骤 7）。具体而言，对于样本中发现的每条日志语句，我们确定其所在的 Python 文件以及出现位置所属的具体类和函数。接下来，我们确定机器学习流水线的哪个阶段与所识别文件（包括其中的日志语句）所包含的实现相关（Amershi 等人，2019年）。为了将日志语句映射到机器学习流水线阶段，我们为机器学习流水线的每个阶段指定一个标签，并为每条已识别的日志语句分配与其所属阶段（如前所确定）对应的标签。这样我们便能回答 RQ2。两位作者并行开展人工分析和标签标注，第三位作者则担任裁判，处理意见不一的模糊案例。对于所有意见不一致的案例，我们都进行了讨论，直至达成共识。为了了解开发者为何在基于机器学习的应用中使用日志语句，并回答研究问题 RQ3，我们人工分析了每条日志语句所在位置的源代码，以确定机器学习从业者在其项目中插入这些日志语句的原因。

3 案例研究结果

本节将介绍三个研究问题的结果。对于每个研究问题，我们都会说明研究动机、回答该研究问题所采用的方法，以及获得的结果。

研究问题1: 基于机器学习的应用程序日志记录实践有哪些特点?

研究动机

许多研究对传统软件应用中的日志记录实践进行了分析（Chen 等人，2017年；Yuan 等人，2012年；Zeng

等人，2019年），揭示了日志记录在软件系统中的重要性。这些研究得出的结论表明，日志记录实践在传统应用中得到了广泛应用。许多研究使用人工智能算法，基于日志文件进行预测和异常检测（Nagaraj 等人，2012年；Xu 等人，2009年）。此外，也有研究致力于帮助开发者做出日志记录决策（Zhao 等人，2017年；Zhu 等人，2015a）。然而，这些研究都没有关注机器学习应用中的日志记录特征。直观而言，机器学习与传统应用开发的编程范式存在显著差异：机器学习从业者采用数据驱动的方法，利用大量数据以及一组参数和超参数来开发和训练模型；相比之下，传统软件开发者采用逻辑驱动的方法，直接实现程序逻辑。因此，直观来看，基于机器学习的应用的日志记录方式会不同于传统应用。因此，在本研究问题中，我们希望研究基于机器学习的应用中的日志记录实践特征。

方法

与以往关于传统软件应用日志记录实践的研究类似（Chen 等人，2017年；Yuan 等人，2012年；Zeng 等人，2019年；Zhu 等人，2015a），我们研究了基于机器学习的应用中日志记录实践特征的以下几个方面。

– **日志库类型**。在数据收集过程中，我们获得了一个包含 `import` 语句的数据集，并对其进行了定性分析，以识别日志库。我们还区分了面向 AI 应用的专用日志记录与通用日志库。为完成这项任务，三位作者对 `import` 数据集进行了分析，以识别潜在的日志库。我们搜索了在线资料（例如，通过谷歌 [18] 搜索引擎），以获取有关疑似日志库的更多信息。识别出这些库后，我们进一步调查了它们的用法。本次调查旨在识别可用于进一步提取相关日志语句的模式。（Chen 等人，2017年）将通用日志库描述为开发者可以设置日志级别的库，例如信息、调试、警告、错误、致命、异常和严重。专用日志库则是为记录特定信息而设计的库；与其他类型的库并发记录日志时，这些信息不会丢失或混杂。我们根据数据收集期间获得的日志语句，将数据集人工标注为两类：通用日志记录和机器学习专用日志记录。发现的日志库汇总见 表2。

[18] `https://bit.ly/3VPCmaM`

表2：基于机器学习的应用中的日志库概述

日志库类型	库	发布方	主要用途
机器学习专用日志记录	超参数	Python 软件基金会	用于记录模型的超参数
	wandb	Weights & Biases	用于数据集版本控制、跟踪超参数，实验跟踪
	TensorFlow	谷歌	在使用以下工具进行开发时，用于记录和汇总机器学习信息 TensorFlow
	日志记录器	NVIDIA	用于在推理阶段记录机器学习信息
	caffe.log	伯克利人工智能研究中心	用于在开发过程中记录机器学习信息 使用 Caffe
	MLflow	Databricks	用于跟踪训练和推理阶段的机器学习实验，并记录指标和超参数
	ml-logger	Python 软件基金会	跟踪机器学习实验
	dllogger	NVIDIA	跟踪机器学习实验
	回调	Keras	跟踪机器学习实验，执行提前停止，
通用日志库	日志记录	Python 软件基金会	在 Python 中记录各种信息
	rospy	开放机器人	在工作时用于记录各种信息 使用机器人操作系统中的 rospy 库
	警告	Python 软件基金会	用于在 Python 项目中记录各类警告信息

– **日志语句密度**。我们在开展本研究时，针对每个项目的最新版本测量日志密度，步骤如下：(i) 克隆项目的最新版本后，使用 cloc[19] 工具评估项目中的代码行总数（源代码行数），同时排除注释和空行。(ii) 然后，我们使用自定义的 AST[20] 解析器 2.1，提取项目中的日志语句，并对其进行定量分析。(iii) 最后，我们使用 $\frac{SLOC}{NL}$ 公式计算每条日志语句对应的代码行数。以往研究也使用过这一指标（Yuan 等人，2012年）。

– **日志语句的详细程度级别**。(Li 等人，2017年) 的研究表明，为日志语句分配正确的日志级别十分重要，而开发者很难确定合适的日志级别。与 (Ze ng 等人，2019年) 类似，我们研究了基于机器学习的应用中日志语句的分布。我们根据研究设计过程（第 2.1 节）中提取的日志语句数据集，按照以下模式评估详细程度级别的分布，即 info、warn、debug、error、except、critical 和 fatal；这些是 Python 官方文档建议的详细程度级别[21]。

为了了解基于机器学习的应用中日志记录实践的特点，我们在 110 个开源项目中研究了上述三个方面。

结果

本节将呈现第一个研究问题的结果，讨论我们的结果和发现，并将其与先前的研究进行比较。我们在研究问题1中发现了三个主要结论。

[19] https://bit.ly/3DgYhjQ
[20] https://bit.ly/3F5bI7S
[21] https://bit.ly/3TJSpEY

发现 ***1***：基于机器学习的应用中普遍使用日志语句，但其普及程度低于 JAVA、C#、C/C++应用，高于 Android 应用。

在研究的 110个研究项目中，78.18%（86 个）至少包含一条日志语句。这表明，基于机器学习的应用中普遍存在日志记录。日志密度用于估算开发者在项目中插入日志语句的可能性；日志密度越小，越有可能在该项目中找到日志语句。我们可以从 图 3 中观察到，与其他项目相比，开发者在 C/C++ 项目中编写日志语句的概率非常高（Yuan 等人，2012年）：平均每 30 代码行就有一条日志语句，而 JAVA 应用中平均每 51 代码行才有一条日志语句。对于基于机器学习的应用，日志密度低于 Android 应用（平均每 290 代码行有一条日志语句，而 Android 应用平均每 479 代码行有一条），但高于 C# 应用（290 行对 58 行）。因此，机器学习从业者在机器学习应用中添加日志语句的概率高于安卓开发者在 Android 应用中添加日志语句的概率，但低于在 JAVA、C# 和 C/C++ 应用中添加日志语句的概率。

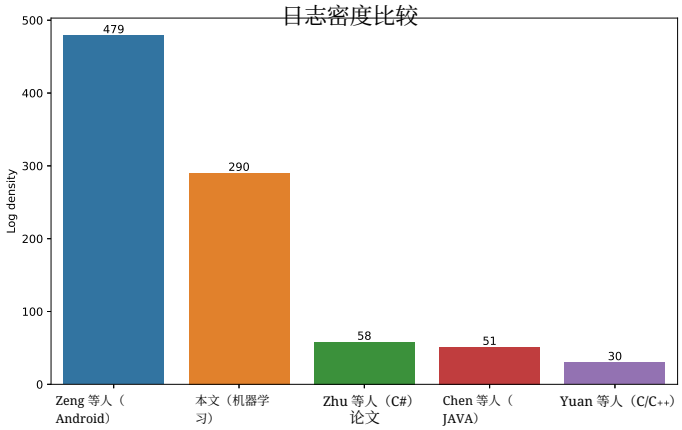


图 3：本文研究的基于机器学习的应用与以往研究中的其他类型应用的日志密度比较。

发现***2***：基于机器学习的应用中的大多数日志语句都使用通用日志库。

图4展示了基于机器学习的应用中，按照日志语句属于通用日志库还是专用日志库进行分类后的分布情况。通用日志语句的使用频率是机器学习组件专用日志语句的11倍。这种主导地位可以解释为：（i）通用日志语句同时用于机器学习组件和非机器学习组件，而基于机器学习的应用中的机器学习专用日志库仅用于机器学习组件。例如，通用日志语句用于机器学习组件（数据处理）中，如下面这段可在 AI-training 中找到的代码片段所示[22]

```
def 转换 ilsvrc2010(目录, 输出目录,                                    -
                                    输出文件名='ilsvrc2010.hdf5',
                                    随机打乱种子 =config.default seed):

    使用 h5py.File(output path, 'w') 作为 f:
        log.info('正在创建 HDF5 数据集......')
        准备 HDF5 文件 (f,          n train,     n valid,     n test)
        log.info('正在处理训练集...... ')
        处理训练集 (f, train, patch, n train,          -          wnid 映射、随机打乱种子)
        log.info('正在处理验证集...... ')
        处理其他集合(f,  -          '有效',     有效、补丁、有效真实标注、n train)          -
        log.info('正在处理测试集...... ')
        处理其他集合(f,  -          '测试',     测试、补丁、测试真实标注,
                                    n 训练  +n 验证)
        日志.信息('完成。')
    返回 (输出路径, )  -


定义 准备元数据（开发工具包归档文件、测试真实标注路径）:                    -

    # 确定文件名数量，以准备适当大小的
    # 数组。
    训练数量  = 整数（同义词集['训练图像数量']。求和（）
    日志.信息('训练集:    {}图像'。格式化（训练数量）          -
    日志.信息('验证集:    {}图像'。格式化（长度（有效真实标注））          -
    log.info('测试集:    {}图像'.format(len(测试真实标注)))-
    总数  = 训练数  + len(验证集真实标注) + len(测试集真实标注)     -
    log.info('总数（训练/验证/测试）:    {}图像'.format(总数))          -
    返回 训练数、验证集真实标注、测试集真实标注、wnid 映射                    -
```

代码清单 1：机器学习组件中使用的通用日志语句代码片段

它们也用于模型训练步骤，如以下 cifar10-human-experiments[23], 中的代码片段所示

```
    如果人工调优:
        logger.info('— 轮次 {}          c10h 训练集          :   {:.4 f} (准确率:   {:.4 f})  | c10h 验证集          :   {:.4 f} (准确率:
                {:.4 f})'。格式(
            轮次、训练损失计量器.avg、训练准确率、损失计量器.avg、准确率))
        日志记录器.info('—                c10h 训练集 c10:   {:.4 f} (准确率:   {:.4 f})  | c10h 验证集 c10:   {:.4 f} (准确率:
                {:.4 f})'.format(
            c10h 训练集 c10 损失计量器.avg、c10h 训练集 c10 准确率、c10h 验证集 c10 损失计量器.avg、      -
                    c10h 验证集 c10 准确率))
    否则 :
        logger.info('轮次 {} 损失 {:.4f} 准确率 {:.4f}'.format(
            轮次, 损失 meter.avg, 准确率))

    已用时  =time.time()  — start
    logger.info('已用时间 {:.2f}'.format(elapsed))
```

代码清单 2：模型训练中使用的通用日志语句代码片段

(ii) 机器学习从业者在机器学习组件中经常使用通用日志语句，而不是专为机器学习组件设计的特定日志语句

[22] https://bit.ly/3F1mdsy
[23] https://bit.ly/3ShI2XW

在他们的应用中。例如，代码清单 1 展示了机器学习从业者使用通用日志语句记录模型参数的情况。使用日志语句“`wandb.log()`”也可以完成同一日志记录任务，该语句专为在模型训练期间记录模型参数而设计。以下代码片段中也有这种情况，该片段可在 BPNN[24]中找到。

```
如果 使用_WANDB:
                wandb.log({"验证均方根误差": np.sqrt(np.mean(验证损失)),"训练均方根误差":
                    np.sqrt(np.mean(损失))},commit=(非args.val 分布外))

                如果 使用_WANDB:
                    wandb.log({"验证均方根误差分布外1": np.sqrt(np.mean(验证损失1))})

                打印("验证分布外1均方根误差 =", np.sqrt(np.mean(验证损失1)))

                如果 使用_WANDB:
                    wandb.log({"RMSE_valOOD3": np.sqrt(np.mean(val 损失3))})

                如果 使用_WANDB:
                    wandb.log({"分布外验证集4均方根误差": np.sqrt(np.mean(验证损失4))},commit=True)
```

代码清单 3：模型训练中使用的机器学习专用日志语句代码片段

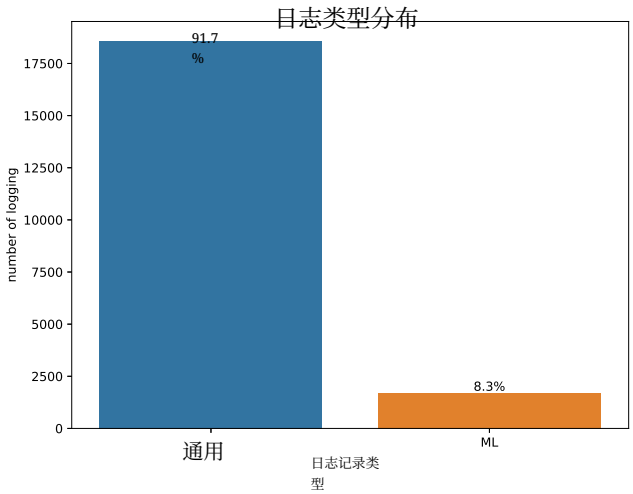


图 4：日志类型分布。

发现 ***3***：基于机器学习的应用中，大多数日志语句的级别为 `info` 和 `warnings`；而在 C/C++ 应用中，大多数日志语句的级别为 `info`；在 Android 应用中，大多数日志语句的级别为 `debug`、 `error` 。

[24] `https://bit.ly/3FbDQWC`

图5 展示了 110个研究项目的日志级别分布。基于机器学习的应用中，超过一半的日志语句属于 `info`和 `warning` 级别。这可以解释为，机器学习从业者在数据和模型管理中经常使用这些日志级别，如以下示例所示：代码清单 1、代码清单 2。其中，信息级别用于模型和数据管理（在数据处理过程中记录模型参数和信息）；警告级别则用于提醒数据预处理操作中的不一致情况。

```
def _包含 NaN（a, NaN 策略='传播'）：
        try：
            包含 NaN =np.nan 位于集合(a.ravel())
        异常 类型错误：
            # 不知道该怎么办。退而求其次，省略 NaN 值，并
            # 发出警告。
            包含 NaN = 否
            NaN 处理策略 = '忽略'
            warnings.warn("无法正确检查输入数组中的 NaN "
                          值。NaN值将被忽略。", 运行时警告)
定义 峰度检验(a, axis=0, NaN 处理策略='传播')：

    如果 n < 20：
        warnings.warn("峰度检验仅在 n>=20 ... 继续 "
                      "无论如何，n=%i" % int(n))

定义 从字典创建数据集( self，  字典, 索引=无, 返回篮子=假, 非初始标记="X")  _  _
                logger.warning(f" [任务: {任务名称}] 无法通过以下方式将标签转换为标识符  标签列表！ "
                               f" \n我们发现标签存在问题 {str(有问题的标签)}")  _
            # TODO 推理标志实现后修改此处
            异常 键错误：
                logger.warning(f" [任务: {任务名称}] 无法通过以下方式将标签转换为标识符  标签列表！ "
                               "\n如果您正在以 *推理* 模式运行：不用担心！"
                               "\n如果您正在以 *训练* 模式运行：请确认您提供了
                                    正确的标签列表给您的处理器，并检查输入中的标签
                                    数据是否正确。")
```

代码清单 4：展示数据处理过程中警告用法的代码片段

(Air-Pollution[25],farn[26])代码清单 4或模型训练 代码清单 5(DeepPavlov[27]) 在 MLcomponent 中，

```
定义 从配置训练并评估模型()  _  _  —> 字典[字符串, 字典[字符串,  浮点数 ]]:

    如果 '训练'  不在 配置中:
        log.warning('缺少训练配置。正在填充默认值')
    训练配置  = config.get('train')

    if '评估目标' not in 训练配置 and ('验证最佳 ' in 训练配置  _  _
                                        or '测试最佳' in  训练配置 ):
        log.warning('"验证最佳" and "测试最佳"  _  参数已弃用。'
                    '请改用"评估目标"  _  列表')

    如果 迭代器 不是None:
        如果 待验证项 不是None:
            如果 评估目标 是 _  无:
                log.warning('"进行验证"  参数已弃用，并将在未来版本中移除。'
                            ' 请改用"评估目标"  _  列表代替')
                评估目标  =['测试' ]
                如果to-validate:
                    评估目标.append('有效')
            否则:
                log.warning('同时指定了"评估目标"和"to validate"参数。' _
                            ' "to validate"已弃用，将被忽略')

    返回res
```

代码清单 5：展示在模型训练中使用警告的代码片段

以及它在非机器学习组件中的用法如下（Django Web[28]）。

[25] https://bit.ly/3SoY9Dh
[26] https://bit.ly/3z0EMJy
[27] https://bit.ly/3TqKwVp
[28] https://bit.ly/3F4EsgT

```
def 方法检查(req, methods):
    try:
        断言 req.method属于methods
    异常 断言错误:
        消息 = '不允许的方法（{方法}）： {路径}'。格式(
            方法=req.method，路径=req.path)
        log.警告(message)
        抛出 ResponseNotAllowed(detail=message)
```

代码清单 6：展示警告在非机器学习组件中用法的代码片段

图 5 所示的分布与以往研究不同。Zeng 等人（2019年）的结果表明，Android 应用中的大多数日志语句都处于 `debug` 和 `error` 级别，而 Yuan 等人（2012年）的结果表明，C/C++应用中的大多数日志语句都处于 `info` 级别。

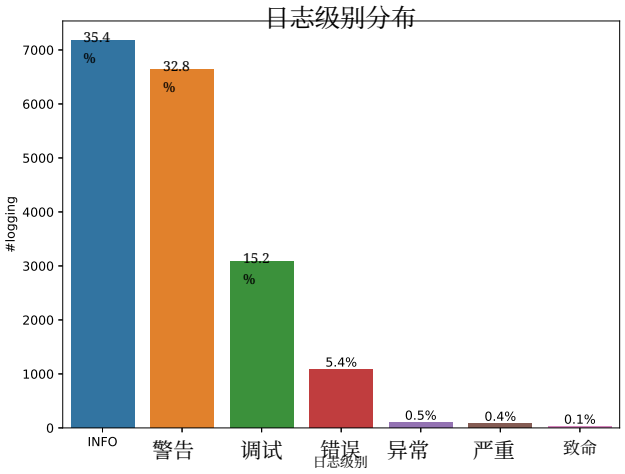


图 5：日志语句级别的分布。

总结： 基于机器学习的应用中的日志记录实践不同于传统应用中的日志记录实践。主要区别之一是，基于机器学习的应用中存在多种日志库，这些库可分为两类：通用日志库和机器学习专用日志库。通用日志库主要用于传统应用，而机器学习专用日志库则专为基于机器学习的应用设计。我们的结果表明，尽管存在众多机器学习专用日志库，通用日志库仍然是基于机器学习的应用中使用最多的日志库。这一发现表明，机器学习从业者可能并不了解机器学习专用日志库的存在，因此可能采用了并非最优的日志记录实践。信息和警告日志级别占主导地位，这可以解释为这些日志级别最常用于在应用中记录信息，具体而言，就是在应用的机器学习组件和非机器学习组件中记录信息。这些结果是理解基于机器学习的应用中日志记录实践的初步探索，也说明有必要进一步研究基于机器学习的应用的良好日志记录实践。

研究问题2： 机器学习流水线的哪些阶段更容易出现日志记录?

研究动机

研究问题1的结果表明，机器学习从业者在开发基于机器学习的应用时广泛使用日志语句。这一发现引出了一个关于整个流水线中日志使用情况的问题：机器学习流水线的哪些阶段包含最多的日志语句？回答这个问题有助于了解开发者在机器学习流水线不同阶段对日志记录的需求，并支持开发用于机器学习系统工程的高效日志库和推荐工具。

方法

为选取用于分析的日志语句，我们按照第2.1小节所述进行分层随机抽样，并获得 ~2条样本日志语句。随后，我们分析样本中每个项目里日志语句所在的代码部分，并将每条日志语句归入以下五个机器学习流水线阶段之一：模型训练、数据收集/加载、数据处理、模型评估和模型部署。我们依据以下线索，将日志语句与机器学习流水线阶段进行匹配：(i) 包含日志语句的文件路径通常会包含一些关键词，例如“train”“data”“load”“load data”“model deploy”“train image”“fit”“tokenizer”“evaluator”

和“data cleaning”。例如，以下日志语句的文件路径便是如此：pykale [29]、Air-Pollution [30]、DeepLearningExamples [31]。(ii) 在文件中，我们特别关注日志语句所在函数的名称、文件中包含的注释以及函数周围的注释。我们还会参考日志语句中的消息。例如，对于以下代码清单[32] 所示的 DeepLearningExamples 样本7，我们将第 2、3、7、16 行的日志语句归类为“模型训练”，将第 21、22、24、30、31 行的日志语句归类为“模型评估”。

```
1  def 训练(n 标记, 截止值, 排名, 本地排名, 每台主机的核心数):
2      tf.logging.info("批次数量 {}".格式(训练记录信息["批次数量"]))
3      tf.logging.info("步骤 {} | 学习率 {:8.9f} "
4                      "| 损失 {:.2f} | 困惑度 {: >7.2f}, 每字符比特数 {: >7.4f}, 令牌/秒 {: >6.0f}".格式(
5                          当前步数,
6                          当前损失,
7                          已获取[-2],
8                          数学.exp(当前损失),
9                          当前损失 / 数学.log(2),
10                         吞吐量))
11             dllogger 数据 = {
12                 '学习率': 已获取[-1],
13                 '训练损失': 当前损失,
14                 '训练困惑度': math.exp(curr loss),
15                 '训练吞吐量': throughput,
16             }
17             dllogger.log(step=int(当前步数), 数据=dllogger 数据)
18             tf.logging.info("模型已保存至路径: {}".format(save path))
19     如果 排名 == 0:
20         tf.logging.info("训练吞吐量: {:>6.0f} 标记/秒".format(meters['训练吞吐量'].avg))
21
22 定义 evaluate(n 标记, cutoffs):
23     如果 FLAGS.max_eval_batch > 0:
24         批次数量 =FLAGS.max eval batch
25     tf.logging.info("批次数量 {}".format(num batch))
26     tf.logging.info("评估 {}".format(eval ckpt path))
27     if (step+1) % (num 批次 // 10) == 0:
28         tf.日志记录.信息(格式字符串格式(步骤+1, 批次数))
29         dllogger 数据 = {
30             '评估延迟': 延迟,
31             '评估吞吐量': 吞吐量,
32         }
33         dllogger.log(step=step+1, 数据=dllogger 数据)
34     tf.日志记录器.info("使用以下配置进行评估: 批次大小 {}, 数学 {}".format(FLAGS.eval 批次大小, "自动混合精度" 如果 FLAGS.amp
35         否则 "单精度浮点数"))
36     tf.logging.info(" | 损失 {:.2f} | 困惑度 {:>7.2f}, 每字符比特数 {:>7.4f}, tok/s {:>6.1f}, 毫秒/批次 {:>4.2f}".format(
37         平均损失, math.exp(平均损失), 平均损失 / math.log(2), meters['评估吞吐量'].avg,
38         meters['评估延迟'].avg))
```

代码清单 7：训练和评估阶段的代码片段

人工分类任务由两位作者完成，卡帕一致性为 0.71，即高度一致。模糊案例在会议上进行了讨论。如果无法在会议上达成一致意见，则将该日志语句标记为未分类案例。总计，我们无法对样本中的 66 条日志语句进行分类。我们将这一结果归因于包含这些日志语句的代码过于复杂，和/或缺少上述关键要素。

结果

在本节中，我们展示第二个研究问题的结果，并重点介绍我们的发现。

[29] https://bit.ly/3zioXy8
[30] https://bit.ly/3TXHbwE
[31] https://bit.ly/3zlGQME
[32] https://bit.ly/3zoQXAd

表3：机器学习流水线不同阶段的日志语句分布

机器学习流水线	描述	示例	% 年龄
模型训练	模型训练包括超参数调优，优化cost函数，拟合数据，and模型选择	logging.info('使用单个进程在 1 个图形处理器上进行训练。') tf.logging.info("[*]训练轮次：%d" % _训练轮次) logging.info(f'训练时间：(elapsed / 60):.2f 分钟') logger.info('轮次[%d] 耗时=%.3f', epoch, (toc - tic)) logger.info(f"正在为以下内容加载预训练文件：', '.join(self.loadables)") dllogger.log(step=(轮次, 每轮步数,_ 批次索引), 数据=dllogger-数据, verbosity=0) dllogger.log(step="参数", 数据='计划轮次':_ 轮次数量、详细程度=0)	35.83%
Data 收集-载入	数据收集/加载包括收集- 从不同来源 和格式收集并读取数据：在线、CSV、JSON、文本、HDF5、结构化 查询语言。	log.info('正在将驾驶数据记录到 %s', self.hdf5 dir)。 logging.info('加载未经身份验证的文件传输协议文件（来自 fromm）'.format(ftpURL, ftpFilePath)) logger.info("正在从 ".format(self.path)) logger.info("数据集加载完成")	14.47%
数据处理	数据处理步骤包括归一化， 化。 数据的转换、验证和特征化 化数据	logger.info(f"正在将标签 repr(saved label) 从索引 "f"index 移动，因为 repr(label) 被放在了它的位置。") tf.logging.warning("'shuffle' 为 false，但输入数据流是 ') logger.info("排名频率： ".format(print dictionary(freq))) logger.info(f"正在调整 type(self) 的输入维度。 __name__" f"从旧维度调整为新维度，以匹配语言模型")'	15.84%
模型评估-tion/验证	在验证数据或 测试数据上评估模型，并选择最佳模型 并验证其正确性	logger.log("将最佳模型保存到： ".format(final best name)) logger.log("最佳模型有：权重。" .format(best model.numel())) logging.info(f'延迟均值：1000.0 * latency data.mean():.2f 毫秒') tf.logging.info('开始评估。') tf.logging.info("***** 正在运行预测*****") logger.info('最终测试准确率 = %.1f%% (N=%d)' % (test accuracy * 100, len(test bottlenecks)))	6.20%
模型部署	将模型或系统部署到 生产环境	logger.info("正在部署模型 " + self.args.triton model name +" 到 format " +self.lib.platform) logger.info("模型检查失败，警告： [", error, "]") logger.info("量化 - 模型忽略") LOGGER.debug("现有 已部署 资源： %s", 存在-记录资源)	3.66%

在分析的 ∼2K 条日志语句中，35.83% 属于模型训练阶段，19.51% 属于项目中不属于任何机器学习组件的代码部分。我们还发现，分别有 15.84%、14.47%、6.20% 和 3.66% 的日志语句属于数据处理、数据加载、模型评估和模型部署。对于样本中 4.50%（即 66 条）的日志语句，我们未能达成一致意见。 图6 展示了所研究的基于机器学习的应用中，不同组件内日志语句所占比例的分布。可以看出，大多数日志语句出现在模型训练组件中，而模型部署阶段的日志语句占比最低。这一结果并不令人意外，因为模型训练阶段产生的代码比例高于其他阶段；为了获得稳定模型，需要优化许多参数，而开发者通常必须记录多条信息，以指导优化过程（训练和重新训练）。在表3中，我们列出了在所研究项目中发现的日志语句示例。这些示例按照所属的机器学习流水线阶段分组。在描述列中，我们介绍了各阶段的特点，特别是列出在将日志语句分类到相应阶段时所考虑的要素。

发现 ***4***：总体而言，整个机器学习流水线的五个阶段都包含日志语句。然而，模型训练阶段的日志语句占比最大，而模型部署阶段的日志语句占比最小。

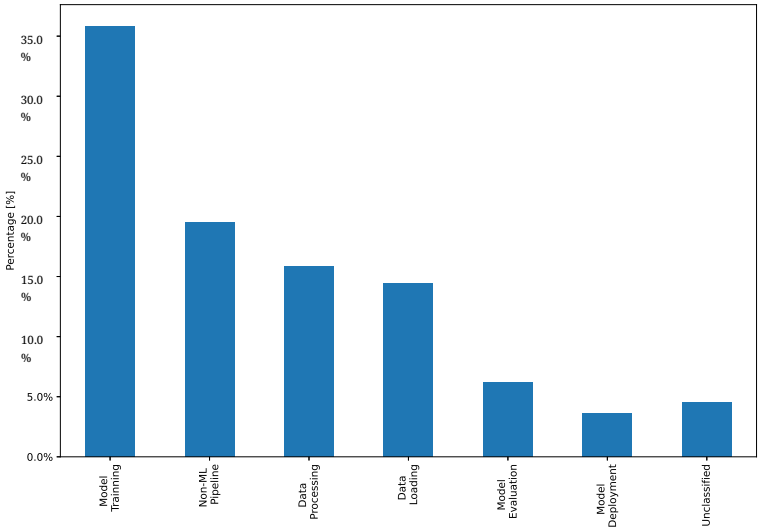


图 6：经分析的日志语句样本中属于机器学习流水线不同阶段的百分比

RQ3： 机器学习从业者为什么使用日志记录?

研究动机

在研究问题1中，我们观察到，基于机器学习的应用中普遍存在日志语句，但其数量少于传统软件应用。我们还观察到，机器学习开发者会同时使用机器学习专用日志库和通用日志库。在这个研究问题中，我们旨在更好地了解机器学习开发者在基于机器学习的应用整个生命周期中对日志记录的需求。具体而言，我们希望确定在机器学习流水线的特定阶段记录的信息类型。深入了解基于机器学习的应用中的日志记录实践，有助于设计高效的日志记录支持工具，也有助于为开发和维护基于机器学习的应用的从业者制定指南。

方法

我们通过定性分析，了解在基于机器学习的应用中使用日志语句的原因。我们从数据集中随机抽取了380条日志语句；对应的置信水平为95%，置信区间为5%。我们详细检查了每条日志语句，包括其日志级别、日志文本、其中的变量以及周围的代码。我们将这些信息与第3节中提到的线索结合起来，以确定所记录的信息类型以及机器学习开发者记录这些信息的原因。接下来，我们根据机器学习开发者使用这些日志语句的原因，以及在项目中引入这些语句时机器学习流水线所处的具体阶段，对日志语句进行分类。本文第一作者和第三作者分别独立审查了每条日志语句。对于所有模糊案例，我们都与第二作者讨论，直至达成共识。

结果

本节报告并讨论我们对基于机器学习的应用中所记录的信息类型，以及机器学习流水线中发生日志记录事件的具体阶段的研究发现。阶段是机器学习流水线某个环节中的重要步骤。

我们的分析揭示了基于机器学习的应用中进行日志记录的两个主要原因：数据管理和模型管理。我们还发现了一个取决于上述两个原因的次要原因，即配置管理。图7 总结了我们归纳出的日志记录原因。图7中的每个信息块图7 都代表在基于机器学习的应用中使用日志记录的一个原因。数据管理和模型管理中的每个实体都代表一个关键阶段（即机器学习流水线某个环节中的重要步骤），而属性则代表被记录的信息类型。通用管理依赖于数据管理和模型管理中的实体，因为它能够记录信息，以确保与训练过程或数据处理过程相关的整个环境都已正确配置。通用管理块中的每个实体都代表使用通用日志库的特定目的，而属性则代表被记录的信息类型。

下面，我们将详细阐述分析中识别出的每个进行日志记录的原因。

数据管理关注数据的生命周期，从数据收集（例如通过 API、网页抓取、S3，以及摄像头和手机等设备）到将数据摄入训练过程，其中包括以下步骤：数据加载、数据并行（即将数据分布到不同节点上，由这些节点并行处理数据）、数据处理和数据流。在输入数据集的整个生命周期中，机器学习从业者通常会使用日志记录来提取重要信息（例如错误信息）并跟踪

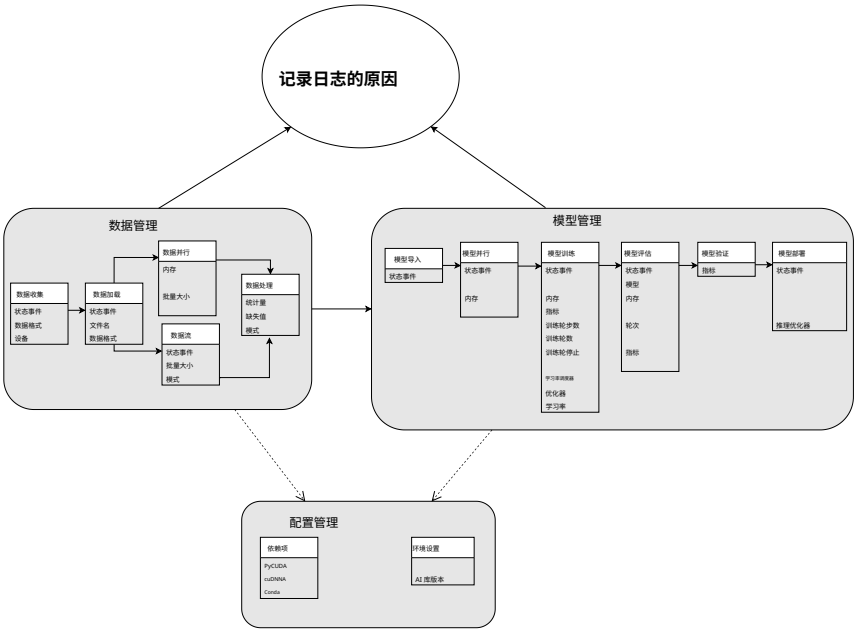


图 7：基于机器学习的应用中记录日志的原因。 实体代表基于机器学习的系统中的不同阶段。属性表示记录的类型或信息。

数据的演变。我们确定了日志记录普遍应用的五个阶段。

– **数据收集**：收集数据时，机器学习从业者通常会记录以下信息，部分示例如列表3中所示**：** **(i)**状态事件日志——其中包含特定事件的信息，例如记录数据，以及在数据收集过程中处理错误数据。这些状态事件日志会在这些事件开始时、进行期间或结束时记录，例如列表 3中的第 1、5、6、7 行。(ii) **数据格式日志**——记录的信息与所收集数据的类型或格式有关，例如第 4、1 行。(iii) **设备日志**——记录的信息与数据采集工具或设备有关，例如第 2、8 行。列表3中的日志语句可在以下项目中找到：DeepLearningExamples[33], DeepCamera[34]。

```
1       log.info('Recording driving data to %s', self.hdf5_dir)
2       log.debug('Grabbing images over ZMQ from %s', conn_string)
3       log.debug('obz_exists? %r should_record? %r', obz is not
↪  None self.should_record),
```

[33] https://bit.ly/3GDstYo
[34] https://bit.ly/3XvpQ10

```
4       logging.info("Converted {} frames in {}".format(num_frames, tfrecord))
5       logging.info("Processed {} records".format(len(tfrecords)))
6       logging.info("Corrupted record {}".format(tfrecord))
7       logging.info("Processing record # {}: {}".format(record_id, record))
8       logging.info('Camera: using Jetson onboard camera')
9
```

列表3：数据收集步骤中的日志语句示例

– **数据加载**：数据收集完成后，在此步骤中，数据会从不同来源（如 S3、目录、硬盘）以不同格式（如 JSON、CSV 等）加载。在此步骤中，机器学习从业者通常会记录：(i) **状态事件日志**——通过此日志，机器学习从业者通常使用状态事件日志，通过记录加载过程的开始（例如第 4、6 行）、数据加载过程中发生的错误（例如第 1 行）、有关文件的信息（例如第 4、3 行）以及数据加载结束（例如第 5、2 行），来监控数据加载过程。(ii) **数据信息日志**——此处记录的信息与文件名和数据格式有关，如图7所示。列表4 给出了我们研究对象中发现的一些日志语句示例，尤其是在以下项目中：cs-ranking[35],deepdrive[36]

```
1       log.error('Could not load %s - skipping - error was: %r', h5_filename,
        ↪  e)
2       log.info('finished loading %s', h5_filename)
3       log.info('data name %s', dataset)
4       logging.info('loading csv')
5       logger.info("Done loading the dataset")
6       log.info('loading %s', h5_filename)
```

列表4：数据加载步骤中的日志语句示例

– **数据并行**：当收集的数据量相较于可用资源（如图形处理器和内存）过大，无法直接加载时，就会执行此步骤。随后，机器学习从业者会进行数据并行化，即根据可用图形处理器或 CPU 的数量划分数据，以加速数据处理并最终加快训练过程。他们通常会使用：(i) **GPU/CPU 日志** ——记录与所用图形处理器或 CPU 数量相关的资源信息（例如第 3、4 行）。(ii)**内存消耗日志** ——机器学习从业者在训练过程中记录内存使用情况的日志语句（例如第 1、2 行）。(ii) **批次日志** ——数据在此处按批次加载，因此会记录批次大小等批次详情（例如第 5 行）。我们在研究对象中识别出的一些日志语句示例见清单 5 ，也可在项目 cs-ranking[37], rtg[38]中找到。

[35] https://bit.ly/3XCGFXQ
[36] https://bit.ly/3U1yhy1
[37] https://bit.ly/3Vkp0lz
[38] https://bit.ly/3EXYwRL

```
1     log.warning(f"Going to buffer data; this may consume all the memory
      ↪  crash. Current usage={max_RSS()[1]}.")
2     log.warning(f"Buffered {self.n_inp_recs:,} records Current memory
      ↪  usage={max_RSS()[1]}")
3     logging.info("device_ids%s", gpu)
4     logger.info("rank%s", cpu)
5     logging.info("sampler size%s", batch_size)
```

列表5：数据并行步骤中的日志语句示例

– **数据流**：此步骤发生在在线学习的机器学习系统中，数据会按顺序输入机器学习算法。当无法在机器学习模型中训练整个数据集，或数据以流式方式到达时，通常会采用此类系统。在此阶段，机器学习从业者必须确保机器学习算法接收到的不同数据流之间保持一致（例如第2行），我们将其称为（i）**模式日志**。为此，机器学习从业者需要了解每个接收到的数据流的大小及其模式，并通过（ii）**状态事件日志**监控每个数据流的接收过程（例如第1、2、3、4、5行）。所有这些信息都能帮助机器学习从业者在数据分布发生变化时进行追踪。列表6 展示了在以下项目中发现的一些相关日志语句：DeepCamera [39], attention-lvcsr[40], 和 attention-lvcsr[41]。

```
1     logger.info("Monitoring on auxiliary data finished")
2     logger.info('No changes in schema detected.')
3     logger.info("sending StopIteration")
4     logger.info("sending {} arrays".format(len(data)))
5     logger.info("Monitoring on auxiliary data started")
```

列表 6：数据流步骤中的日志语句示例

– **数据处理**：此步骤涉及处理数据的属性和标签，以从数据中提取有意义的信息。在处理过程中，机器学习从业者通常会记录（清单 7）：**(i)缺失信息日志**——与缺失值、缺失标签相关（例如第 1、6、7 行）。**(ii)统计日志**——此日志语句包含局部估计量（均值、中位数、最小值、最大值）和变量估计量（方差、标准差）（例如第 3、4 行）。**(iii) 模式日志**——此处的日志语句包含有关某些数据属性模式的信息（例如第 5 行）。这些示例可在以下主题中找到：csrank [42]、Air-Pollution。[43]

[39] https://bit.ly/3XE6ren
[40] https://bit.ly/3XlSuBD
[41] https://bit.ly/3i8ulOw
[42] https://bit.ly/3Ox7tUZ
[43] https://bit.ly/3Oz2svs

```
    logger.info("Missing values {}: {}".format( col, np.isnan(arr).sum() / len(arr))
    logger.info("Sampled instances {} objects {}".format(X.shape[0], X.shape[1]))
    log.info(f"mean:{mean}, std:{std:.4f} ;; low:{low:.4f} high:{high:.4f} ;; invert:{invert}")
    logger.info("Min {}: Max {}".format(np.nanmin(arr), np.nanmax(arr)))
    logging.info('FORMAT --dateformat: {}'.format(FORMAT))
    tf.logging.fatal('Label does not exist %s.', label_name)
    logger.info("Missing values {}: {}".format( col, np.isnan(arr).sum() / len(arr))
```

清单 7：数据处理步骤中的日志语句示例

模型管理。 模型管理涉及模型的开发，包括开发、评估、验证并将其发布到生产环境。这一阶段的一大难点是跟踪为寻找最佳模型而进行的所有实验，并使模型能够更好地泛化到新数据上。机器学习从业者通常会反复进行多项实验，才能找到最佳模型。确定哪些参数或实验促成了最佳模型是一项繁琐的任务，也可能耗费大量时间，尤其是在手动操作时。解决这一问题的一种方法是记录所有相关信息。记录所有相关信息有助于机器学习从业者复现或比较以往的实验，进而选出最佳模型。我们确定了模型管理过程中的六个重要步骤，在这些步骤中，机器学习从业者会记录相关信息。

– **模型导入**：此步骤包括以不同格式导入模型或预训练权重，机器学习从业者通常会在此过程中记录 **状态事件日志** ，以确保模型导入过程正确完成，例如清单8中的示例，该示例来自 AI-Training[44] 项目。

```
    logger.info(f"Loading pretrained files for: {', '.join(self.loadables)}")
    log.info('Downloading weights %s', folder)
    logging.info("Model loading done")
    logger.info("loading model from {}".format(self.path))
```

列表 8：模型导入步骤中的日志语句示例

– **模型并行**：当模型大小受到内存限制，无法在单个图形处理器上训练大型模型时，就会采用这一步骤。因此，机器学习从业者通常会将模型拆分到多个图形处理器设备上。在这一步骤中，机器学习从业者

[44] https://bit.ly/3Vg7BdM

通常会记录 (i) **状态事件日志** - 其中包含记录训练过程开始和结束信息的日志语句（例如第 5 行）。(ii) **内存消耗日志** - 与训练阶段内存消耗相关的日志语句。(iii) **GPU/CPU 日志** - 有关所使用的图形处理器和 CPU 设备的信息（例如第 1、2、3、6、7 行）。清单 9展示了此类日志语句的一些示例。

```
1  logging.info('Training in distributed mode with multiple processes, 1
   ↪ GPU per process. Process %d, total %d.'% (args.rank, args.world_size))
2  logger.info('device: {}'.format(device))
3  logger.info("device: {} n_gpu: {} distributed training:
4  ↪ {}".format(device, n_gpu, bool(args.local_rank != -1)))
   logger.info('begin training on multiple GPU')
5  logger.info("device: {} n_gpu: {}".format(device, n_gpu))
6
7  logging.info('Gradient averged for the rank of {}'.format(rank))
```

列表 9：模型并行步骤中的日志语句示例

– **模型训练**：在此步骤中，机器学习从业者通过调整模型的诸多超参数和优化函数，使用数据训练机器学习模型。此步骤耗时且耗费资源，并且具有非确定性，因此机器学习从业者需要在训练过程中跟踪大量信息，并通过日志语句记录这些信息（例如，清单 10 中所示的示例）。（i）**时间日志**——评估训练时间或完成一个轮次所需的时间，例如第 15 行。（ii）**内存消耗日志**——评估内存消耗，例如第 14 行。（iii）**超参数日志**——记录训练过程中的超参数，例如第 2、6、7 行。（iv）**指标日志**——记录训练过程中的指标，例如第 18、13 行。（v）**模型大小日志**——记录模型大小，例如：第 4、14、16 行。（vi）**状态事件日志**——记录状态事件，以便跟踪训练过程的开始、可能发生的错误、超参数更新以及训练过程的结束，例如第 5、8、13 行；还可记录与机器学习算法学习过程结束相关的关键信息，例如第 1、8 行。在以下研究对象中可以找到一些日志语句：BPNN [45]、attention-lvcsr [46]、AutoDL-Projects [47]、AutoDL-Projects [48]。

```
1  logger.log("Early stop the pre-training at {:}".format(iepoch))
2  log.info(f"New learning rate dividor = {self._learning_rate_cur_div}")
3  log.info('pooling_0 shape: %s' % pooling_0.shape)
4  logger.log("The base-model has {:}
   ↪ weights.".format(base_model.numel()))
5  logger.log("[ONLINE] [{:03d}/{:03d}] loss={:.4f},
   ↪ score={:.4f}".format(idx, len(env), future_loss.item(), score )
```

[45] https://bit.ly/3V2Y2PE
[46] https://bit.ly/3EWJziG
[47] https://bit.ly/3tRdHpm
[48] https://bit.ly/3Xqnnoo

```
6       logger.log("scheduler  : {:}".format(scheduler))
7       logger.log("optimizer  : {:}".format(optimizer))
8       logger.info("Initializing the training algorithm")
9       log.info('Did not improve on the {} of {}'.format(m_name,
        ↪  self.score_best))
10      logging.info('rank %s: failing epoch %s batch %s', rank, epoch, batch)
11      wandb.log({"RMSE_training": np.sqrt(np.mean(losses))})
12      logger.log("TRAIN [{:}] loss = {:.6f}".format(iter_str, loss.item()))
13      logger.error("Error occured during training." + error_message)
14      logger.log("FLOP = {:} MB, Param = {:} MB".format(flop, param))
15      logging.info(f'Training time: {(elapsed / 60):.2f} minutes')
16      logger.log("The model size is {:.4f}
        ↪  M".format(xmisc.count_parameters(model)))
17      wandb.log({"RMSE_val": val_RMSE, "RMSE_training": training_RMSE})
```

清单 10：模型训练步骤中的日志语句示例

– **模型评估**：此步骤是模型开发过程的一部分，包括寻找能够泛化数据的最佳模型，以及评估模型在验证数据或测试数据上的性能。在此阶段，机器学习从业者通常会记录相关信息，以便追踪整个评估过程：**(i)状态事件日志** — 通常用于记录评估过程的开始、发生的错误以及结束，例如第 5 行。**(ii) 最佳模型日志** — 记录微调阶段获得的最佳模型信息，例如第 1 行。**(iii) 内存消耗日志** — 记录评估过程中的内存消耗，例如第 2 行。**(v) 指标日志** — 记录每一轮次的性能度量（指标、损失），例如第 8、3 行。**(iv)模型大小日志** — 记录模型大小，例如第 7 行。**推理时间日志** — 记录模型推理预测结果所用的时间，例如第 6 行。列表11 列出了本研究对象在此阶段收集的日志语句示例，可在 AutoDL-Projects 中找到[49], DeepLearningExamples[50]。

```
1       logger.log("Save the best model into {:}".format(final_best_name))
2       logger.log( ""Finish training/validation in {:} with Max-GPU-Memory of
↪  {:.2f} MB  and save final checkpoint into {:}"".format( ,
↪  convert_secs2time(epoch_time.sum, True)
3       tf.logging.info('%s: Step %d: Validation accuracy = %.1f%% (N=%d)'↪  %(datetime.now(),
4        i, validation_accuracy * 100,len(validation_bottlenecks)))
5       logger.info("***** Running evaluation *****")
6       logger.info("time for inference {} perf {}".format(eval_end -
↪  eval_start  num_examples * 100 / (eval_end - eval_start))) ,
7       logger.log("FLOP = {:} MB, Param = {:} MB".format(flop, param))
8       logger.log("{:} {:} epoch={:03d}/{:03d} :: Train [loss={:.5f},
↪  acc@1={:.2f}%  acc@5={:.2f}%] Valid [loss={:.5f}, acc@1={:.2f}%, ,
↪  acc@5={:.2f}%]".format( time_string(), need_time, epoch, total_epoch, train_loss, train_acc1,
 train_acc5, valid_loss,↪
↪  valid_acc1  valid_acc5 ) ,,
```

[49] https://bit.ly/3GKwJ8J

[50] https://bit.ly/3VmRc7k

清单 11：模型评估步骤中的日志语句示例

– **模型验证**：此步骤旨在评估评估阶段获得的模型在真实数据上的精确率和性能。为此，机器学习从业者会记录**指标**，用作衡量其机器学习系统性能的指标，如列表12所示。

```
1     logger.info('Final test accuracy = %.1f%% (N=%d)' % (test_accuracy *
      ↪  100  len(test_bottlenecks))) ,
```

清单 12：模型验证步骤中的日志语句示例

– **模型部署**：在此阶段，为了跟踪该阶段发生的所有情况，机器学习从业者通常会记录以下信息：**(i) 状态事件日志** - 这是与以下过程相关的日志语句：通过应用图优化和层合并，将模型转换为更轻量的中间表示，例如第 1、2、3、6 行。**(ii)GPU/CPU 日志** - 在模型预测期间会记录 CPU 和 GPU 等设备信息。**(iii)时间延迟日志** - 在生产环境中，快速交付推理结果至关重要。因此，机器学习从业者会记录一次推理的执行时间，例如第 7 行。**(iv) 优化器日志**– 可使用不同的优化器将机器学习模型转换为针对实时推理进行优化的更轻量表示，例如第 4、6 行。**(iv) 精度日志** – 优化器启用的精度是机器学习从业者在模型投入生产时记录的一项重要信息，例如第 5 行。清单 13 展示了我们在研究对象中收集的、对应模型部署步骤的日志语句示例。

```
1     logger.info("model check failed with warning: [", error, "]")
2     logger.warning("Warning during onnx.checker.check model in quantized
↪  model ignored")
3     logging.info('Total node count before and after TF-TRT conversion:',
↪  num_nodes  '->'  len(frozen_graph.node)) „
4     logger.info('TRT node count:',len([1 for n in frozen_graph.node if
↪  str(n.op)'TRTEngineOp']))
5     tf.compat.v1.logging.info("Precision = %s", "fp16" if FLAGS.amp else↪  "fp32")
6     tf.compat.v1.logging.info('Converting graph using↪  TensorFlow-TensorRT...')
7     tf.compat.v1.logging.info("Total Inference Time W/O Overhead = %0.2f
↪  for Sentences = %d"  predict_time_wo_overhead  num_sentences) ,,
```

清单 13：模型部署步骤中的日志语句示例

配置管理 步骤并未在机器学习流水线中明确呈现，但对于确保机器学习组件随着时间推移始终按预期运行至关重要。配置管理

活动通常包括管理特殊设备上的配置或依赖项，以及管理确保机器学习组件按预期运行所必需的库。通过分析，我们识别出了与以下配置管理活动相关的日志：

– **依赖配置日志**：当应用程序的机器学习组件依赖特定库时，通常会执行此操作，以确保应用程序的机器学习模块能够正常运行。因此，机器学习从业者需要记录所用重要库的相关信息。清单 14 展示了此步骤中发现的一些日志语句。这些语句可见于我们研究的对象中：DeepLearningExamples[51]，deepdrive[52]。

```
log.info('Installed UEPy python dependencies')
warnings.warn('CVM does not support memory profile, using Stack VM.')
warnings.warn("PyCUDA import failed in theano.misc.pycuda_init")
logging.info("Using torch DistributedDataParallel. Install NVIDIA Apex
↪  for Apex DDP.")
```

清单 14：依赖配置步骤中的日志语句示例

– **环境设置日志**：机器学习应用运行的环境不同于传统应用；事实上，机器学习应用通常需要更多资源或特殊设备，例如 CPU 和图形处理器，才能正常运行。机器学习从业者通常会确保应用运行所需的环境中至少具备一定数量的资源。因此，系统会记录关键资源的相关信息（例如清单 15中的示例）。以下研究对象中也有一些示例：DeepPavlov[53],AutoDL-Projects[54]，AI-Training[55]。

```
logger.log("cuDNN   Version  :
↪  {:}".format(torch.backends.cudnn.version()))
logger.log("PyTorch Version  : {:}".format(torch.__version__))
logger.warning("This recipe needs the sox-io backend of torchaudio")
logger.log("CUDA available   : {:}".format(torch.cuda.is_available()))
logger.log("CUDA GPU numbers : {:}".format(torch.cuda.device_count()))
"logger.log(""CUDA_VISIBLE_DEVICES : {:}""
.format( os.environ[""CUDA_VISIBLE_DEVICES""] if
↪  ""CUDA_VISIBLE_DEVICES"" in os.environ else ""None"")"
caffe.log('Using devices %s' % str(gpus))
_logger.warning("We are not able to detect the number of CPU cores." "
↪  We disable openmp by default.")
_logger.info('Conda mkl is not available: %s', e)
```

清单 15：环境设置步骤中的日志语句示例

[51] https://bit.ly/3Eyc1WB
[52] https://bit.ly/3Ex9tZ5
[53] https://bit.ly/3TZ4mXi
[54] https://bit.ly/3tRawy3
[55] https://bit.ly/3ACmajS

发现*5*：机器学习从业者使用日志语句记录数据管理（例如，在数据收集中记录数据格式）、模型管理（在模型训练中记录超参数）和配置管理（例如，记录所使用的资源或库）中的重要信息。我们的观察结果为机器学习从业者改进机器学习日志记录提供了指导，也为未来改进机器学习日志记录实践提供了启示（例如，提供机器学习日志记录库，以便在基于机器学习的应用的整个生命周期中记录不同类型的重要信息）。

4 效度威胁

本节将讨论我们的研究方法和研究结果可能面临的效度威胁。

外部效度

本文用于回答不同研究问题的研究对象是来自 GitHub 的开源机器学习项目。这些项目的选择可能面临以下威胁：

我们的结果和研究发现可能不适用于使用其他语言（例如 JAVA、C# 或 R）而非 Python 编写的机器学习项目，因为在本研究中，我们的分析主要针对研究对象中使用 Python 编写的代码部分。因此，未来有必要研究使用其他编程语言编写的机器学习项目，因为使用这些语言的机器学习从业者在机器学习项目中可能采用不同的日志记录实践。然而，由于 Python 被认为是机器学习应用开发的通用语（Dilhara 等，2021年），我们相信，本研究有助于深入理解基于机器学习的系统中的日志记录实践。

构念效度

我们的研究在构念效度方面面临的威胁，与提取日志语句时可能出现的错误有关。

为了避免提取开发者注释掉的日志语句，我们在标准的 Python AST 解析器（该解析器广泛用于 Python 代码的静态分析（D'Souza et al.,2016年; Dilhara et al., 2021年））之上开发了一个静态代码分析器。该静态代码分析器可在我们的复现包中获取[56]，它只提取未被注释掉的语句，使我们能够避免收集开发者注释掉的日志语句。

[56] 本研究使用的脚本和数据文件可在线获取，点击此处即可找到：`http://bitly.ws/yr6c`

内部效度

为回答 RQ2，我们手动将日志语句映射到相应的机器学习流水线阶段。随后，在 RQ3 中，我们手动分析了这些日志语句，以确定所记录的信息类型并创建分类体系。然而，我们的手动分析受执行分析者主观判断的影响，这对结果的内部效度构成了威胁。为减轻这一威胁，本文的三位作者共同进行了手动分析，他们在机器学习系统工程领域拥有丰富的行业和学术背景。两位作者先进行手动分析；如有分歧，我们会与第三位作者进行小组讨论，直至达成共识。我们认为，这种方法降低了分析中出现误报的可能性。不过，我们仍希望未来能对本研究进行复现和扩展。本研究使用的所有数据和脚本均可在我们的复现包中获取[57]

5 相关工作

本节将介绍并讨论日志记录实践相关工作的两个领域：（i）关于日志记录特征的研究，以及（ii）关于日志记录决策的研究。

（i）**日志记录实践特征：** 许多研究致力于刻画日志记录实践。（Yuan 等人，2012年）首次对使用 C/C 编写的四个开源项目中的日志记录实践进行了实证研究++ ，发现日志记录十分普遍。（Chen 等人，2017年）通过分析 21 个使用 JAVA 编写的项目开展了类似研究。该研究是一项复现实验，旨在将（Yuan 等人，2012年）针对使用 C/C 编写项目所得出的研究结果推广到使用 JAVA 编写的项目。其结果表明，使用 JAVA 编写的应用与使用 C/C++编写的应用在日志记录实践方面存在差异。（Zeng 等人，2019年）研究了 1,444 个 Android 项目中的日志记录实践，并将其研究结果与先前工作的结果进行了比较。然而，之前的研究均未关注基于人工智能的应用中的日志记录实践。本文填补了这一知识空白。

（ii）**日志记录决策：** 日志记录决策研究旨在帮助开发者决定在哪里引入日志语句，以及应设置何种详细程度。这些研究使用人工智能算法来预测应在哪里记录日志或记录什么内容。（Zhu 等人，2015b）提出了一种使用机器学习技术的“学习日志记录”框架，旨在帮助开发者在开发过程中决定在哪里添加日志语句。此外，Li 等人（2020年）采用基于深度学习的方法，帮助开发者在代码块级别做出日志记录决策。Li 等人（2017年）提出了一种序数回归模型，可准确提供

[57] http://bitly.ws/yr6c

开发者添加日志语句时关于日志级别的建议；Li 等人（2021年）也提出使用基于序数的神经网络来建议日志级别。最近，Mastropaolo 等人（2022年）提出了 LANCE 框架，利用深度学习生成完整的日志语句。这显然表明，人工智能算法已被用于辅助传统应用的开发者做出日志记录决策，但尚无研究协助人工智能从业者在基于人工智能的应用中做出日志记录决策。因此，我们的工作为未来研究提供了良好的起点。

6 结论

开发者已将日志记录实践纳入良好的编程实践中。日志通常能帮助开发者在运行时诊断程序，从而减少维护工作。传统软件系统中的日志记录实践已成为许多研究的主题，研究对象包括移动应用、JAVA 应用和开源应用。据我们所知，本文首次尝试研究基于机器学习的应用中的日志实践。我们研究了来自 GitHub 的 110 个开源基于机器学习的应用程序。研究得出了以下发现：基于机器学习的应用普遍使用日志，但其普及程度低于 JAVA、C#、C/C++ 应用，高于 Android 应用。大多数日志语句属于 `INFO` 和 `WARNING`级别。此外，我们发现基于机器学习的应用使用两类日志库：通用日志库和机器学习专用库。然而，尽管存在机器学习专用库，通用日志库仍是机器学习应用中最常用的库。为了确定基于机器学习的应用中哪个机器学习流水线包含最多日志语句，我们进行了定性与定量分析。研究结果表明，大多数日志语句出现在模型训练阶段，而模型部署阶段的日志语句占比最小。此外，我们观察到，机器学习从业者使用日志语句记录与数据管理（例如，在数据收集过程中记录数据格式）、模型管理（在模型训练过程中记录超参数）和配置管理（例如，记录所用资源或库）相关的重要信息。本文的贡献如下：

– 据我们所知，这是首项对基于机器学习的应用中的日志记录实践进行定量和定性分析的研究。– 我们识别出了基于机器学习的应用中使用的通用日志库和机器学习专用日志库。– 我们识别出了从业人员使用日志的机器学习阶段，以及他们记录的信息类型。我们的研究结果为机器学习从业者改进机器学习日志记录提供了指导，也为未来改进机器学习日志记录实践的工作提供了见解（例如，提供机器学习日志记录库

以便在不同的机器学习阶段记录不同类型的信息）。

总体而言，我们的研究结果凸显了开发更多机器学习专用库的必要性，以支持基于机器学习的应用的开发。我们还需要开展更多研究，以加深对机器学习系统工程背景下日志记录需求和挑战的理解。

利益冲突

作者声明不存在利益冲突。

数据可用性声明

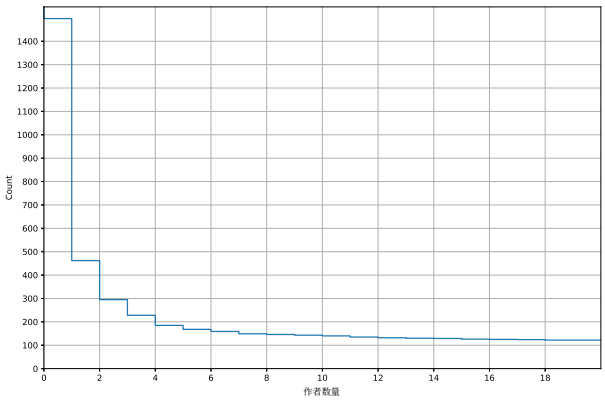
当前研究期间生成和/或分析的数据集可在 [foalem]代码仓库中获取： [https://github.com/foalem/ML-logging- paper]。

参考文献

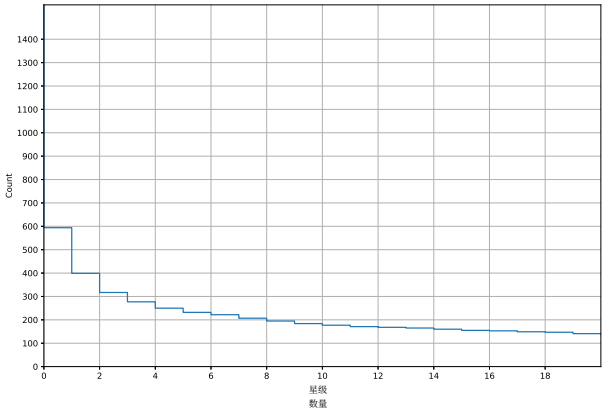
Amershi S, Begel A, Bird C, DeLine R, Gall H, Kamar E, Nagappan N, Nushi B, Zimmermann T（2019）机器学习软件工程：案例研究。载于：2019年 IEEE/ACM 第 41 届国际软件工程会议：软件工程实践（ICSE-SEIP），IEEE出版社，第 291–300 页。Chen B 等（2017）基于 Java 的开源软件项目日志记录实践的特征分析——Apache 软件基金会中的一项复制研究。《实证软件工程》22(1)：330–374。Dilhara M, Ketkar A, Dig D（2021）理解软件 2.0：机器学习库使用情况及其演进研究。ACM《软件工程与方法论汇刊》（TOSEM）30(4)：1–42。Ding Z, Li H, Shang W（2022）Logentext：使用神经机器翻译自动生成日志文本。载于：2022年 IEEE 国际软件分析、演化与再工程会议（SANER），第 349–360 页，DOI 10.1109/SANER53432.2022.00051。D'Souza AR, Yang D, Lopes CV（2016）用于改进 Python API 推荐的集体智能。载于：2016年 IEEE 第 16 届源代码分析与操作国际工作会议（SCAM），IEEE出版社，第 51–60 页。Kernighan BW, Pike R（1999）《程序设计实践》。Addison-Wesley Longman Publishing Co., Inc.，美国。Li H, Shang W, Hassan AE（2017）开发者应该为新的日志语句选择哪种日志级别？《实证软件工程》22(4)：1684–1716。

Li Z, Chen TH, Shang W（2020）我们应该在哪里记录日志？研究并建议代码块中的日志记录位置。载于：2020年 IEEE/ACM 国际自动化软件工程会议（ASE）第 35 届会议，第 361–372 页。Li Z, Li H, Chen TH, Shang W（2021）Deeplv：使用基于序数的神经网络推荐日志级别。载于：2021年 IEEE/ACM 第 43 届国际软件工程会议（ICSE），第 1461–1472 页，DOI 10.1109/ICSE43902.2021.00131。Mastropaolo A, Pascarella L, Bavota G（2022）利用深度学习生成完整的日志语句。arXiv 预印本 arXiv:220104837。Munaiah N, Kroh S, Cabrey C, Nagappan M（2017）为工程软件项目整理 GitHub。《实证软件工程》22(6)：3219–3253。Nagaraj K, Killian C, Neville J（2012）系统日志的结构化比较分析，用于诊断性能问题。载于：第 9 届 USENIX网络系统设计与实现研讨会（NSDI 12），第 353–366 页。Openja M, Majidi F, Khomh F, Chembakottu B, Li H（2022）研究在 Docker 上部署机器学习项目的实践。载于：Staron M, Berger C, Simmonds J, Prikladnicki R（编）《EASE 2022：2022年国际软件工程评估与评价会议》，瑞典哥德堡，2022年6月13日至15日，ACM，第 190–200 页，DOI 10.1145/3530019.3530039，网址 https://doi.org/10.1145/3530019.3530039。Xu W, Huang L, Fox A, Patterson D, Jordan MI（2009）通过挖掘控制台日志检测大规模系统问题。载于：ACM SIGOPS 第 22 届操作系统原理研讨会论文集，第 117–132 页。Yamane T（1967年）《统计学：入门分析》。技术报告。Yang T, Agrawal V（2016年）日志文件异常检测。CS224d，2016年秋季。Yuan D, Park S, Zhou Y（2012）开源软件日志记录实践的特征分析。载于：2012年国际软件工程会议第 34 届会议（ICSE），IEEE出版社，第 102–112 页。Zeng Y, Chen J, Shang W, Chen THP（2019）研究移动应用日志记录实践的特征：以 F-Droid 为例。《实证软件工程》24(6)：3394–3434。Zhai H, Casalnuovo C, Devanbu P（2019）Python程序中的测试覆盖率。载于：2019年 IEEE/ACM 第 16 届软件仓库挖掘国际会议（MSR），IEEE出版社，第 116–120 页。Zhao X, Rodrigues K, Luo Y, Stumm M, Yuan D, Zhou Y（2017）Log20：在指定开销阈值下全自动优化日志打印语句的位置。载于：第 26 届操作系统原理研讨会论文集，第 565–581 页。Zhu J, He P, Fu Q, Zhang H, Lyu MR, Zhang D（2015a）学习记录日志：帮助开发者做出明智的日志记录决策。载于：2015年 IEEE/ACM 第 37 届国际软件工程会议，IEEE出版社，第 1 卷，第 415–425 页。Zhu J, He P, Fu Q, Zhang H, Lyu MR, Zhang D（2015b）学习记录日志：帮助开发者做出明智的日志记录决策。载于：第 37 届国际软件工程会议论文集——第 1 卷，IEEE出版社，ICSE ’15，第 415–425 页。

附录 A 图



(a) 基于作者数量的累积频率曲线



(b) 基于星级数量的累积频率曲线

图 8：星标数量和作者数量的累积频率曲线