

机器学习如何改变软件开发实践?

Zhiyuan Wan, Xin Xia, David Lo and Gail C. Murphy

摘要——为系统添加一种学习的能力，这在本质上增加了系统的不确定性。考虑到越来越流行将机器学习集成到系统中，**我们想知道添加内容如何改变软件开发实践**。我们对来自四大洲 26 个国家的 14 名受访者和 342 名受访者进行了定性和定量的研究，**以得出机器学习系统的发展与非机器学习系统发展之间的显著差异**。我们的研究揭示了软件工程各个方面(例如，需求、设计、测试和过程)和工作特征(例如，技能多样性、问题解决和任务标识)的显著差异。基于我们的研究结果，我们强调未来的研究方向，并为从业者提供建议。

关键词——软件工程，机器学习，实践者，实证研究

1. 介绍

机器学习(ML)在过去的 30 年里取得了巨大的进步，从一个实验室里的新奇事物发展成为一名广泛应用于商业的实用技术。在人工智能领域，机器学习已经成为开发有用软件系统的首

选方法，这些软件系统可用于计算机视觉、语音识别、自然语言处理、机器人控制和其他应用。机器学习功能可以通过多种方式添加到系统中，包括**带有 ML 组件的软件系统和提供 ML 功能的 ML 框架、工具和库**。一个广泛的趋势已经出现：开发和部署 ML 系统¹相对快速和便宜，但是随着时间的推移，由于技术债务[19]，维护它们是困难和昂贵的。**ML 系统¹具有非 ML 软件系统的所有问题，以及一组额外的 ML 特定问题**。例如，概率建模为机器提供了一个框架，使其能够从观察到的数据中学习，并推断出能够做出预测的模型。不确定性在概率建模[14]中起着至关重要的作用：观测数据可以与各种模型保持一致，因此在数据不确定的情况下，哪种模型是合适的。对未来数据和行动未来后果的预测也是不确定的。为了解决 ML 的具体问题，最近的研究致力于构建测试[26]、[30]、[36]、[39]的工具，调试机器学习代码[16]、[27]、[28]，以及创建支持 ML 系统[3]、[6]开发的框架和环境。尽管

¹ 在本文中，除非另有说明，否则我们使用 ML 系

统来指代提供 ML 功能的软件框架、工具和库，或者包括 ML 组件的软件系统。

做出了这些努力，软件从业人员仍然难以操作和标准化使用 ML²的系统的软件开发实践。软件开发实践的操作化和标准化对于高质量和可靠的 ML 系统的成本效益开发至关重要。机器学习如何改变软件开发实践？为了系统地研究其影响，我们进行了定性和定量研究的混合，以调查由机器学习引起的软件开发的差异。我们首先对 14 名在 ML 和非 ML 领域都有经验的软件从业者进行开放式访谈，他们平均有 7.4 年的软件专业经验。通过访谈，我们定性地调查了受访者所感知到的差异，并得出了 80 个描述差异的候选语句。我们通过三次焦点小组讨论进一步改进了候选陈述，并对来自四大洲 26 个国家的 342 名软件从业者进行了调查，从数量上验证了我们的访谈中发现的差异。调查对象从事不同的工作，即、开发(69%)、测试(24%)和项目管理(7%)。我们调查了以下研究问题：

RQ1. 将 ML 合并到系统中如何影响软件开发实践？

开发 ML 系统与开发非 ML 系统是否不同？有什么不同呢？如果开发 ML 系统确实不同于非 ML 软件开发，那么过去的软件工程研究可能需要扩展，以便更好地解决开发 ML 系统的独特挑战；

以前的工具和做法可能不适用于 ML 系统的开发；软件工程教育者可能需要教授开发 ML 系统的不同技能。我们的研究发现，在 ML 和非 ML 开发之间的软件工程实践中存在一些统计上显著的差异：

- 需求**:在 ML 系统开发过程中收集需求需要进行更多的初步实验，这就需要性能的可预测下降。

- 设计**:ML 系统的详细设计更耗时，而且往往以密集迭代的方式进行。

- 测试和质量**:为 ML 开发收集测试数据集需要更多的工作；良好的测试性能并不能保证 ML 系统在生产中的性能³。

- 过程和管理**:数据的可用性限制了 ML 系统的能力；数据处理对整个过程的成功更为重要。

RQ2. 应用心理学中的工作特征，如技能多样性、工作复杂性和问题解决能力，在将 ML 纳入系统时是如何变化的？

当实践者在他们的软件开发实践中涉及 ML 时，软件开发的环境(例如，技能多样性、工作复杂性和问题解决)是如何变化的？我们的研究发现，在 ML 和非 ML 开发之间的工作特性有几个统计上的显著差异：

- 技能多样性**:ML 开发需要大量的数学、信息论和统计学知识。

² <https://twitter.com/AndrewYNg/status/1080886439380869122>

³ 在本文中，除非另外提到，我们使用性能来指代模型性能

- **工作复杂性和问题解决**: ML 实践者对于构建系统有一个不太清晰的路线图。
- **任务标识**: 为 ML 开发的任务制定准确的计划要困难得多。
- **互动**: ML 从业者与客户沟通的频率往往较低。

基于这些发现，我们提出了我们的受访者所讨论的确定差异背后的原因——**需求和算法中的不确定性，以及数据的重要作用**。我们还提供了关于初步实验、重现性和性能报告的作用的实践经验，并强调了一些研究途径，如持续性能度和调试。

本文的贡献如下：

- 我们进行了定性和定量的混合研究，以调查由于机器学习的影响，软件实践和从业者工作的差异；
- 我们为研究人员提供了实际意义，并概述了未来的研究方向。

本文其余部分的结构如下。第 2 节简要描述了 ML 开发的过程和概念。在第三部分，我们详细描述了我们的研究方法。在第四部分，我们展示了我们的研究结果。在第 5 节中，我们将讨论我们的研究结果的含义以及对我们的研究结果有效性的任何威胁。第六部分简要回顾了相关工作。第 7 节得出结论并概述今后工作的途径。

2. 背景

机器学习系统的开发是一项多方面、复杂的任务。多种形式的 ML 开发过程：[2]、[11]、[12]、[35] 已经被提出。这些流程共享几个共同的基本步骤：环境理解、数据管理、数据建模以及生产和监视。

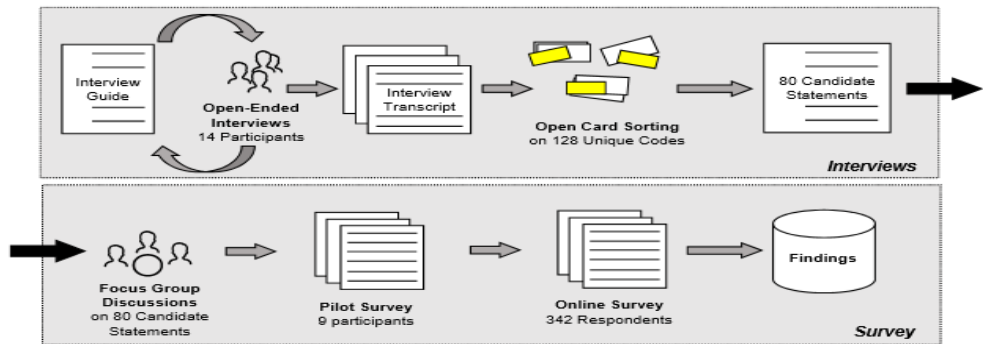
在环境理解步骤中，ML 实践者确定可以从机器学习和可用数据中获益的业务领域。ML 实践者将与利益相关者沟通机器学习能不能管理预期。最重要的是，ML 实践者通过在特定的应用程序环境中进行初步实验来构建和确定开发任务的范围。

数据管理步骤包括来自不同数据源的数据收集、数据预处理、培训、验证和测试数据集创建。由于数据通常来自不同的来源，ML 从业者应该将数据拼接在一起，并通过数据预处理处理丢失或损坏的数据。要为监督学习技术创建合适的数据集，数据标签被要求为每条记录分配地面真相标签。

数据建模步骤包括特征工程、模型训练和模型评估。特征工程是将给定的数据转换为易于解释的形式的活动，包括特征提取和机器学习模型的选择。在模型培训期间，ML 实践者使用所选的特性选择、培训和调优机器学习模型。模型调优包括调整参数和识别当前模型或前面步骤中的潜在问

题。在模型评估中，从业者使用预定义的评估度量来评估测试数据集上的输出模型。

谈结果中归纳出 80 个候选陈述，并进行了三次焦点小组讨论，将我们的候选陈述减少到 31 个最终的调查陈



图表 1 研究方法

在生产和监视步骤中，ML 实践者将模型导出为预定义的格式，并通常创建一个 API 或 Web 应用程序，将模型作为端点。ML 实践者还计划使用更新的数据对模型进行再培训。对模型性能进行持续监视，以发现错误或意外结果，并监视输入数据，以确定它们是否随着时间发生变化，从而使模型失效。

我们使用上面的 ML 开发过程和相

3. 方法

我们的研究方法采用定性和定量相结合的方法，如图 1 所示。我们收集了来自不同来源的数据⁴：(1)我们采访了 14 名在 ML 开发和非 ML 开发方面都有经验的软件从业者；(2)我们从访

述；(3)我们调查了 342 名受访者，具体情况如下：为了保持参与者的匿名性，在分析数据之前，我们将构成个人信息(PII)的所有条目进行了匿名化，并在整个研究过程中进一步将别名考虑为 PII(例如，将受访者称为 P1 - P14)。

3.1 面试

3.1.1 协议

第一作者对 14 名同时具有 ML 开发和非 ML 开发经验的软件从业者进行了一系列面对面的访谈。每次面试都需要 30-45 分钟。Guest 等人认为，对一个同质群体进行 12 到 15 次访谈就足以达到饱和。当我们的采访接近尾声时，我们观察到一种饱和状态。面试是半结构化的，使用的是面试指南⁵。

⁴ 访谈、焦点小组及问卷调查均获有关院校检讨委员会批准。参与者被告知，我们需要他们的意见；没有明确提到隐私和敏感资源。

⁵ 在线采访指南：<https://drive.google.com/file/d/1Z0XwbSKY6zPnu0EzG1zFMJ3DIERYD8YG>

该指南包含主题和问题的一般分组，而不是预先确定的特定集合和问题顺序。

采访分为四个部分。在第一部分中，我们询问了一些关于受访者在 ML 开发和非 ML 开发中的经验的人口统计学问题。我们涵盖了编程、设计、项目管理和测试等各个方面。

在第二部分中，我们问了一个开放式的问题，关于受访者注意到 ML 开发和非 ML 开发之间有什么不同。这部分的目的是让面试者自由地谈论差异，而不是让面试官干扰他们的回答。

在第三十四部分，我们给面试者列出了两个话题，让他们讨论他们没有明确提到的话题。其中一个列表来自于软件工程知识体 (SWEBOK) [7] 指南，它由 10 个知识领域组成，例如，软件设计和软件测试。另一个列表来自应用心理学中的一般工作特征 [18]，包括 21 个工作特征，如技能多样性和问题解决能力。我们选择了 SWEBOK，以确保软件工程主题得到全面的讨论，并确保我们涵盖了广泛的潜在差异。在第三部分中，受访者被要求从两个列表中选择三个话题进行讨论。第四部分，采访者从之前的两次访谈中所讨论的话题中选择三个话题，以确保

话题的覆盖面。

每次面试结束时，我们都会向面试者表示感谢，并简要地告诉他/她我们下一步的计划。

在访谈过程中，每位受访者平均谈论了 6 个话题，其中他/她分享了他/她认为的 ML 开发和非 ML 软件开发之间的差异 (最小值:1, 最大值:12, 平均值:6.6, 标准差:3.2)。受访者提到的主题包括:SWEBOK:需求 (9 受访者)、SWEBOK:设计 (6 受访者), SWEBOK:建筑 (10 受访者) SWEBOK:工具 (7 受访者), SWEBOK:测试 (9 受访者), SWEBOK:质量 (5 受访者), SWEBOK:维护 (4 受访者), SWEBOK:过程 (8 受访者), SWEBOK:配置管理 (3 受访者) 工作:技能多样性 (10 受访者), 工作:工作复杂性 (5 受访者), 工作:问题解决 (4 名受访者), 工作:任务识别 (7 名受访者), 工作:自主性 (1 名受访者), 工作:相互依赖 (4 名受访者), 工作:组织外部互动 (1 名受访者)。

3.1.2 参与者选择

我们从中国杭州的三家 IT 公司，即阿里巴巴、邦盛⁶、恒天⁷，招募了有 ML 系统和非 ML 系统经验的全职员工。邦盛是一家拥有 400 多名员工的技术提供商，为金融业和反欺诈产品开发

⁶ <http://www.bsfit.com.cn>

⁷ <http://www.hengtiansoft.com/?lang=en>

实时风险控制系统。恒天是一家拥有 2000 多名员工的外包公司，专注于欧美企业(如道富银行、思科、路透社)的外包项目。我们通过给每个公司的联系人发电子邮件的方式招募受访者，他们负责将我们的研究结果传播给他们的同事。如果志愿者愿意无偿参与这项研究，他们会通知我们。通过这种方法，14 名志愿者联系了我们，他们多年来有着不同的经验。在本文的其余部分，我们将这 14 位受访者表示为 P1 到 P14。这 14 位受访者平均拥有 7.6 年的专业经验(最小值: 3、最大值:16, 中位数:6, 标准差:4) 包括 2.4 年 ML 系统开发经验(最小值: 1, 最大值: 5, 中位数:2, 标准差: 1.6)和 5.2 年非 ML 软件开发经验(最小值: 2, 最大值: 11, 中位数:4.5, 标准差: 2.6)。表 1 总结了受访者中认为自己拥有某一特定职位的“广泛”经验(相对于“没有”和“有一些”经验)的人数。

表格 1 在特定岗位拥有丰富经验的受访人数

Role	Machine Learning	non-Machine-Learning
Programming	5	6
Design	5	3
Management	2	2
Testing	2	3

3.1.3 数据分析

我们采用主题分析[8]对访谈记录进行处理，步骤如下:转录编码。最后一次访谈结束后，我们将访谈录音转录下来，通过对访谈文本的审阅，对

访谈内容有了较为全面的了解。第一作者阅读转录本，使用 NVivo 定性分析软件[1]对访谈进行编码。为了保证代码的质量，第二作者对第一作者创建的初始代码进行了验证，并提出了改进建议。在整合了这些建议之后，我们总共生成了 295 张卡片，其中包含了每次编码面试的 15 到 27 张卡片。将具有相同单词或含义的代码合并后，我们总共有 128 个惟一的代码。我们注意到，当我们的采访接近尾声时，从采访记录中收集的代码达到了饱和。新的代码不再出现;代码列表被认为是稳定的。

打开卡片分类。然后，两位作者分别分析了代码，并将生成的卡片按主题相似程度分类出潜在主题(如 LaToza 等人的研究[24]所示)。排序过程中出现的主题并不是预先选定的。然后,我们使用科恩的 Kappa 测量[10]来检验两个标签之间的一致性。两个贴标签机之间的总体 Kappa 值为 0.78, 这表明两个贴标签机之间有很大的一致性。在完成贴标过程后，两位贴标签者讨论了他们的分歧，达成了一个共同的决定。为了减少两位作者在整理卡片形成最初主题时的偏见，他们都对最后一组主题进行了审查并达成了一致意见。最后，我们推导出 80 个候

选语句来描述这些差异。

3.2 焦点小组

为了当 ML 被合并到软件系统中时, 我们想要深入研究最有可能不同的语句将调查的重点放在一个可管理的大小上。为了确定 80 个候选语句中哪个具有这种特性, 第一作者进行了三次焦点小组会议。每次焦点小组会议持续 1.5 至 2 小时, 有 3 名参与者参加。这 9 位参与者来自中国不同的 IT 公司 (如百度、阿里巴巴和华为), 他们在 ML 和非 ML 开发方面都有丰富的经验。他们被告知我们的研究目的, 并同意将焦点小组的结果用于研究目的。

在焦点小组会议期间, 第一作者浏览了 80 条候选语句, 并提出了以下问题: “与非 ML 开发相比, ML 开发的语句更真实吗?” 根据反馈, 我们删除了 7 个陈述, 其中参与者不理解 ML 和非 ML 开发之间的差异, 或者不认为存在差异。此外, 我们观察了 42 个陈述, 其中超过一半的焦点小组参与者认为 ML 和非 ML 开发之间没有明显的差异。最后, 我们确定了一个包含 31 个语句的列表。

3.3 调查

3.3.1 协议

调查的目的是量化由广泛的软件

从业者反应出的 ML 和非 ML 软件开发之间的差异, 我们遵循 Kitchenham 和 Pfleeger 的个人意见调查指南[23], 并使用匿名调查来提高[37]回复率。被调查者的选项包括不回答或不理解特定问题的描述。我们包括这个选项, 以减少受访者提供任意答案的可能性。

招聘的受访者。调查的参加者知悉我们的研究目的, 并同意将调查结果用作研究用途。

为了从 ML 和非 ML 人群中招募受访者, 我们将调查范围广泛地扩展到来自世界各地的公司。为了从不同的背景中获得足够多的受访者, 我们采用了以下策略来招募受访者:

- 我们联系了来自不同国家和 IT 公司的专业人士, 请他们帮助在他们的组织内传播我们的调查。我们向来自世界各地的亚马逊、阿里巴巴、百度、谷歌、恒田、IBM、英特尔、IGS、柯达、联想、微软、摩根士丹利等公司的联系人发送电子邮件, 鼓励他们完成调查, 并将其传播给部分同事。通过遵循这一策略, 我们的目标是招募来自不同组织的行业受访者。

- 我们向 1831 名从业者发送了一封带有调查链接的电子邮件并邀请他们参与调查, 这些从业者为 GitHub 上 18 个排名最高的机器学习知识库 (如

TensorFlow 和 PyTorch)做出了贡献。我们的目标是通过将 GitHub 贡献者发送到机器学习存储库来招募除业内专业人士之外的开源实践者。我们选择了这组潜在的调查对象以收集 ML 从业人员的回答作为对比;如果 ML 应答者的应答与非 ML 应答者有显著差异,这就为建立 ML 与非 ML 开发之间的差异提供了定量证据。此外,选择 18 个高评级机器学习存储库的原因是:贡献者可能有两种类型:ML 框架/工具/库 (ML FTL)的实践者和 ML 应用⁸(ML App)的实践者。我们不确定软件差异中的高差异是否会压倒 ML 和非 ML 差异。在这些邮件中,有 8 封邮件收到自动回复,通知我们收件人不在。我们不需要从受访者那里收集任何身份信息。

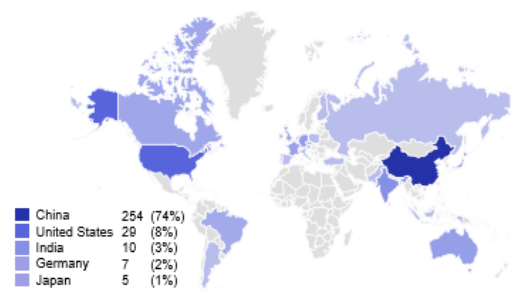
3.3.2 调查设计

我们在调查中获得了以下信息 (完整的问卷可以在网上作为补充材料获得⁹):

人口统计数据: 我们收集了受访者的人口统计资料,以便(1)筛选可能不了解我们调查的受访者(如职位相关程度较低的受访者), (2)对结果进行分组(如开发人员和测试人员;ML 从业者 和非 ML 从业者)。具体来说,我们问了

这样一个问题:“你目前从事的主要产品领域的最佳描述是什么?”,并提供选项: (1)ML 框架/系统/库、(2)ML 应用程序、(3)非 ML 框架/系统/库、(4)非 ML 应用程序和(5)其他。受访者从提供的选项选择一个作为他们的主要产品区域。根据他们的选择,我们将调查对象分为 5 组。

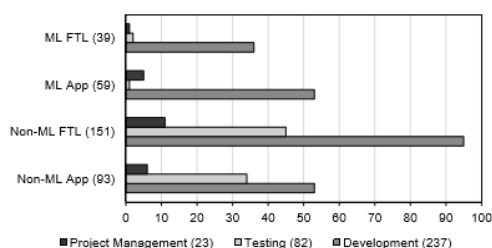
我们总共收到 357 份回复。我们排除了 10 个受访者的回答,他们的工作角色既不是开发、测试也不是项目管理。这些受访者所从事的工作有研究员(5),学生(3)(1)网络基础设施专家和大学教授(1)。我们还排除了五种回答,这些回答的受访者选择其他主要产品领域并将其主要产品领域指定为:生态学(1)、物理学(1)或多个产品领域的组合,这些产品领域似乎不以生产与商业相关的软件产品为目标(3)最后,我们得到了 342 个有效的回答。



图表 2 受访者居住的国家。肤色越深,在该国居住的受访者就越多。由这一说法展示的受访者最多的前五个国家。

⁸ 带有 ML 组件的软件系统。

⁹ 网上问卷:<https://drive.google.com/file/d/>



图表 3 受访者人口统计数据。这个数字表示每个人口组的数量。

如图 2 所示, 342 名受访者分布在四大洲的 26 个国家。受访者居住的前两个国家是中国和美国。受访者的专业经验年限从 0.1 年到 25 年不等, 平均为 4.3 年。我们的调查对象分布在不同的人群中(工作角色和产品领域), 如图 3 所示。

从业人员的看法。我们提供了 31 个最终陈述的列表, 并要求从业者以李克特 5 分制(强烈不同意、不同意、中立、同意、强烈不同意)对每个陈述做出回应。为了将受访者的注意力集中在调查中的某一特定领域, 他们被明确要求根据自己对所指定的主要产品领域的经验来对每一项陈述进行评分。

我们对一小部分不同于我们的受访者、焦点小组参与者和调查对象的从业人员进行了初步调查。我们得到了以下方面的反馈: (1) 调查的长度是否合适, (2) 调查的术语的准确性和可理解性。我们对初稿作了一些小修改, 根据收到的反馈意见进行调查并制作

出最终版本。注意, 从试点调查中收集到的回答不包括在本文给出的结果中。


为了支持来自中国的受访者, 我们在公布调查结果前将调查结果翻译成中文。我们选择用中文和英文进行调查, 因为中文是最常用的语言, 而英语是一种国际通用语言。我们希望我们的调查对象中有很多人能流利地使用这两种语言中的一种。我们仔细翻译了我们的调查, 以确保在我们的调查中不存在英语和汉语术语的歧义。此外, 我们根据试点调查的反馈, 通过提高清晰度和可理解性, 对翻译进行了改进。

3.3.3 数据分析


我们研究了我们的参与者的李克特反应的分布, 并使用 Wilcoxon 秩和检验比较了不同组的参与者的分布, 即, ML 与非 ML, 以及 ML 框架/工具/库与 ML 应用程序。我们在第 4.3 节中报告了全部结果; 在 4.1 和 4.2 节中, 我们将受访者的意见与调查结果联系起来, 参考调查陈述, 如: [S1]。我们按陈述在调查中出现的顺序编号, 从 S1 到 S31。我们对每一项指标进行了标注, 以确定它们在统计上是否具有重要意义, 如下:


•【✓ S1】 ML 开发与非 ML 开发的显著性差异, 证实了受访者的回答, ML 框


架/工具/库的开发与 ML 软件应用的开发没有显著性差异;

- 【S1】ML 开发与非 ML 开发有显著性差异, ML 框架/工具/库开发与 ML 软件应用开发有显著性差异;

- 【S1】无显著差异;

- 【S1】ML 开发与非 ML 开发之间存在显著差异, 但与受访者的回答相反。

- 【S1】ML 发展与非 ML 发展有显著差异, 但与受访者的回答相反; 开发 ML 框架/工具/库与开发 ML 软件应用有显著差异。

- 【S1】ML 开发与非 ML 开发无显著差异; 但开发 ML 框架/工具/库与开发 ML 软件应用之间存在着显著的差异。

其他结果在理论上是可能的, 但在我们的调查结果中没有出现。

4. 结果

在本节中, 我们将报告根据访谈主题分组的结果。当受访者对某一特定话题没什么可说的时候, 我们就把几个话题合在一起。在某些情况下, 为了保护被采访人的隐私, 我们会对部分引文进行匿名处理。

4.1 RQ1. 软件工程实践中的差异


4.1.1 软件需求

几乎每个受访者都对 ML 系统的需求与非 ML 系统的需求之间的差异做出

了强有力的陈述。从本质上讲, ML 系统的需求通常是数据驱动的——与特定应用程序环境的现有大规模数据紧密耦合。

受访者指出, ML 系统的需求比非 ML 系统更不确定【S1】。正如 P9 所指出的, 给定“机器学习系统通常旨在改进或加速决策过程(组织或公司中的高管)”, 而不是详细的功能描述, 需求通常包括关于应用机器学习系统后的目标的概念描述。由于机器学习系统的需求是数据驱动的, 不同的数据会导致不同的需求。即使对于相同的数据, 正如 P1 和 P6 所建议的, 对数据和不同应用程序环境的不同理解也会导致不同的需求。然而, 关于数据和应用程序环境的先验知识在一定程度上带来了决定论。P6 给出了一个具体的例子, 说明先验知识如何帮助理解数据和应用环境:

存在一种先验知识, 称为情景先验知识。例如, 我们知道在欺诈检测的应用中会出现数据不平衡的问题。这是因为好人总是比坏人多。我们也知道, 在网络广告领域, 转化率通常看起来很低, 是有限制的。

在非 ML 软件系统中, 数量度量代替了功能首先, ML 系统的高层架构设计是相对固定的【S4】。正如 P3 所

总结的，ML 系统的体系结构通常包括数据收集、数据清理、特性工程、数据建模、执行和部署。相反，非 ml 软件系统的体系结构设计是一个更具创造性的过程，它实现了软件组件的各种结构分区，并生成行为描述(例如，活动图和数据流图)(P12)。由于数据量大，分布式体系结构风格被广泛应用于 ML 系统。分布式体系结构风格通常会导致体系结构和详细设计的复杂性。需求，它构成了 ML 系统的大部分需求。正如 P4 所指出的，将利用不同类型的定量度量来定义需求，例如，精确度、精密度、召回率、F 度量和标准化折现累积收益(nDCG)。顺序度量可以来自业务涉众的“船长的召唤”(P6)，也可以由项目经理通过用户研究收集(P5)。正如 P4 所说，定量测量的目标分数可能因应用程序的不同而有所不同。在一些安全关键领域，ML 系统的精度是非常重要的。因此，出于安全考虑，预计会有更高的定量措施得分。P5 也表示赞同：对于网上购物推荐系统，量化措施相对不那么严格，较低的措施是可以容忍的。

与非 ML 系统相比，ML 系统的需求通常涉及大量的前期实验【✓ S2】。正如 P6 所指出的，业务涉众可能建议利用一些新兴的机器学习算法来解决他

们的业务问题。其结果之一是，它要求需求专家具有强大的机器学习技术背景。另一个结果是，需求验证过程涉及大量的前期实验。这些初步实验是由软件工程师进行的，旨在验证和选择各种候选机器学习算法。P8 解释说，例如，A、B 和 C 算法可能都适用于特定的应用程序环境，但是性能与实际数据密切相关。在进行初步实验之前，不能验证需求。

需求应该考虑 ML 系统性能的可预测下降【✓ S3】。正如 P6 所指出的，大多数 ML 系统在生产一段时间后可能会出现性能下降。P6 给出了一个例子：在一个欺诈检测应用程序中，对手总是试图找到方法来逃避检测策略。因此，ML 系统有两个固有的需求。首先，ML 系统应该是降解敏感的，即能够察觉到性能下降。其次，一旦出现性能退化，ML 系统需要有相当大的能力来适应这种退化，要么向学习算法提供新数据，要么使用新数据训练全新的模型。

我们将在后面的部分中讨论，与非 ML 系统相比，ML 系统的数据驱动和大规模特性对它们的开发方式有几个影响。

4.1.2 软件设计

受访者多次提到，ML 系统和非 ML

软件系统的设计在几个方面有不同的侧重点。

首先, ML 系统的高层架构设计是相对固定的【✗ S4】。正如 P3 所总结的, ML 系统的体系结构通常包括数据收集、数据清理、特性工程、数据建模、执行和部署。相反, 非 ML 软件系统的体系结构设计是一个更具创造性的过程, 它实现了软件组件的各种结构分区, 并生成行为描述(例如, 活动图和数据流图)(P12)。由于数据量大, 分布式体系结构风格被广泛应用于 ML 系统。分布式体系结构风格通常会导致体系结构和详细设计的复杂性。

其次, 与非 ML 软件系统相比, ML 系统不太重视组件中的低耦合【S5】。尽管 ML 系统中的不同组件具有独立的功能, 但它们是高度耦合的。例如, 数据建模的性能取决于数据处理。正如 P14 所指出的, “‘垃圾输入和垃圾输出’——我将把 40%的时间花在数据处理上, 因为我发现糟糕的数据处理可能会使任何潜在的有效(机器学习)模型失败……我将数据处理分为多个步骤, 每一步都可以使用现有的库。

第三, 与非 ML 软件系统相比, ML 系统的详细设计更加灵活【S6】。P1 指出, 数据建模可以包含数十到数百种机器学习算法的候选项, 这意味着有

足够的搜索空间。P6 回应说即使对于相同的机器学习算法, 不同的应用上下文也可能导致数据维度的差异, 进而导致机器学习模型的变化。

因此, ML 系统的详细设计将是费时的, 并且是以迭代的方式进行的【✓S7】。为了设计一个有效的模型, 软件工程师往往会进行大量的实验。

4.1.3 软件建设和工具

受访者报告了 ML 系统和非 ML 软件系统在编码实践方面的一些差异。首先, 与非 ML 软件系统相比, ML 系统的编码工作量较低【S8】。ML 系统中的编码通常包括数据处理(例如转换、清理和编码)、特征分析(例如可视化和统计测试)和数据建模(例如超参数选择和模型培训), 而不是为在非 ML 软件系统中实现特定功能而编码。P14 指出了用于数据处理和数据建模的有用框架和库的可用性。这些框架和库帮助开发人员加速编码过程。为了获得更好的性能, 开发人员可以扩展这些框架或库, 使其适合自己使用。其次, 与非 ML 软件系统相比, ML 系统之间和内部的代码重用很少【S9】。一个原因是 ML 系统常常非常重视性能。然而, ML 系统的性能在很大程度上取决于数据; 数据在不同的应用程序环境中是不同的。因此, 项目特定的性能调优是

必要的。

在非 ML 软件系统中进行调试的目的是定位和修复代码中的 bug【S10】。与非 ML 软件系统不同, ML 系统中的调试旨在提高性能。ML 系统的性能通常“直到数据模型最终确定的最后一刻”才会被察觉或评估(P13、P14)。有效地确定数据模型在 ML 系统的构建中起着重要的作用。然而, 数据建模涉及多个迭代的训练回合。考虑到数据量大, 如果采用完整的数据, 每一轮训练可能需要很长时间, 几天或几周。使用完整的数据来为每一轮训练模型是不可行的。因此, 有几位受访者提出了一个实用的数据建模过程(P4, P6, P13): 您需要构建从小型到大型不同大小的几个培训数据集。您从小型数据集开始培训模型。除非你达到了可接受的结果, 否则你将进入一个更大的范围。最后, 一路向上, 你会发现一个令人满意的模型。

虽然这个过程提高了训练效率, 但是不完整的训练数据可能会导致中间结果的不准确, 从而导致模型的偏差。

受访者提到, ML 系统和非 ML 软件系统在调试实践上存在差异。非 ML 软件系统的调试实践通常是通过断点一步一步地执行程序。对于 ML 系统, 特

别是深度学习软件系统, “调试的目的是使开发人员的想法更容易地转化为代码”。P6 给出了一个关于“动态计算图”的具体例子: 以前, 开发人员更喜欢使用 PyTorch, 主要是因为它支持动态计算图。“动态计算”是指模型按照您编写的顺序执行。就像我在建立一个神经网络一样, 我会把每一层都加起来。然后每一层都有一些权衡; 例如, 我想添加一个实现规范化的操作符层。(如果调试工具不支持动态计算图形,) 在神经网络编译成模型并输入实际数据之前, 我无法评估这个添加是否正确。动态计算图可以立即对样本数据进行调试, 并帮助我快速验证我的想法。

然而, 在动态计算图上进行调试存在一些不足。一旦数据量非常大, 每一层的计算都需要很长时间才能完成。这延迟了进一步的层的构建。此外, 受访者还提到, 在 ML 系统的调试中, 创造力似乎很重要【S11】。一定程度上是因为 ML 系统具有广泛的模型调优搜索空间。

调查指出了 ML 系统和非 ML 软件系统之间 bug 的几个不同之处。首先, ML 系统不像非 ML 软件(P11)【S12】那样有那么多驻留在代码中的 bug。相反, 错误有时隐藏在数据中(P10)。P1

给出了一个最近的例子，错误地使用训练数据和测试数据导致了令人难以置信的良好性能，但也指出了一个 bug。正如 P4 所建议的，“还需要注意数据模型的泛化”。其次，当考虑到数据时，ML 系统有特定类型的 bug。正如 P11 所述，在集成这两个框架时，两个框架之间的数据维度顺序不匹配可能会导致错误。第三，与非 ML 软件系统相比，单个失败案例很难帮助诊断 ML 系统中的 bug。正如 P13 所解释的，有时候，ML 系统的开发人员只是通过“盯着他们代码的每一行并试图思考为什么它会导致错误”来发现错误。

4.1.4 软件测试与质量

尽管软件质量在 ML 和非 ML 系统中都很重要，但是测试的实践似乎有很大的不同。一个显著的差异存在于测试结果的重现性。与非 ML 软件系统相比，ML 系统的测试结果难以重现，因为有很多随机性的来源【S13】。P8 解释说：（在 ML 系统中）随机性使测试变得复杂。你有随机的数据，随机的观察顺序，随机初始化你的权重，随机的批次反馈到你的网络，随机优化在不同版本的框架和库…虽然可以进行初始化，但是修复批处理可能会降低性能，因为您必须关闭许多框架所做的并行批处理生成。还有一种方法可以在一

次迭代后冻结或删除权重，解决了权重初始化的随机性。

另一个区别存在于测试方法和结果输出。ML 系统的测试实践通常包括多次运行算法并收集性能要求的变化【S14】。正如 P12 解释说，机器学习软件测试实践的主要目的是验证表征性能的量化措施。然而，机器学习算法是随机的。因此，我们通常使用 k-fold 交叉验证来进行测试。

如 P9 所示，测试输出应该是一个范围，而不是单个值。被调查者表示，与非 ML 系统相比，ML 系统的测试用例生成更具挑战性。自动化测试工具不像非 ml 系统那样频繁使用【✗ ✓ S18】。测试数据集对测试用例的质量至关重要。收集测试数据集是劳动密集型【✓ ✓ S15】。P5 指出，如果应用程序用于一般用途，并且人类用户知道正确的答案，那么标记任务可以外包给组织外部的非技术人员。这些自动化方法或工具需要更多的细节。然而，偏差可能会通过方法或工具引入测试数据集，从而影响性能和可泛化性。正如 P8 所说我们（开发人员）有时使用我们自己构建的算法或模型为测试用例生成预期的结果。矛盾的是，这可能会引入偏差，因为这就像我们为代码定义了接受标准一样。此外，对于某些 ML 系统，

生成可靠的测试 oracle 有时是不可行的。P6 在异常检测应用环境中给出了一个具体的例子：客户端给我们一个测试数据集，并告诉我们数据集包含标记的异常情况。然而，我们无法确切地知道数据集中有多少异常情况，因为一些异常可能没有在数据集中被识别和标记。良好的测试结果并不能保证 ML 系统在生产中的性能【✓ S17】。生产中的性能在很大程度上取决于训练数据集和传入数据的相似性(P6)。

在 ML 系统中，测试结果中性度量度的“过低”和“过高”分数都表明存在 bug【S16】。P1 给出了一个最近的例子，一个初级开发人员在他的数据模型中获得了 F1 99% 的分数。事实上，在仔细研究了数据集之后，我们发现训练数据集和测试数据集之间存在很大的重叠。一些受访者报告了测试 ML 系统的几种具体策略(P13)：我们可以使用一个简单的算法作为基线，例如，一个随机算法。如果基线在数据集中执行得很好，那么数据集中可能存在 bug。如果我们的算法执行得比基线差，那么我们的代码中可能会有一些 bug。

4.1.5 软件维护和配置管理

受访者认为，与传统软件系统相比，ML 系统的维护工作可能需要更少

的精力【S19】。一个原因是，与非 ML 软件系统不同，ML 系统的性能会随着时间的推移而下降(P4 和 P7)。为了不断提供健康的结果，ML 系统应该支持“自动”维护。一旦性能下降，ML 系统就被设计用来感知性能下降，并使用最新出现的数据以联机/脱机方式训练新的数据模型。P6 建议，我们有时定义机器学习系统状态的所谓“健康因素”或定量指标。它们与特定的应用程序环境相关联。这些指标有助于机器学习系统在特定的应用程序环境中感知其性能。

受访者表示，与非 ML 软件相比，ML 系统的配置管理涉及的内容更多【S20】。原因之一是机器学习模型不仅包括代码，还包括数据、超参数和参数。开发 ML 系统需要快速的实验和迭代。模型的性能也会随之变化。要找到实现最佳性能的这些部分的最佳组合，需要配置管理跟踪不同的模型和相关的权衡、算法选择、体系结构、数据和超参数。P4 解释说：通常情况下，我目前训练的模型表现很差。我可能会回到以前的模型，并调查原因……云中的数据会随着时间而变化，包括我们用来训练模型的数据。模型可能会随着数据的发展而退化。我们可以查看当前的数据，并将它们与以前的数

据进行比较，以了解为什么会发生退化。

因此，与非 ML 软件相比，ML 系统的配置管理变得更加复杂。除了代码和依赖项之外，数据、模型文件、模型依赖项、超参数还需要配置管理。模型检查点和数据将占用大量空间。正如 P8 所指出的，机器学习框架用精确的数字决定论来换取性能，“依赖的框架可能会随着时间的推移而发生变化，有时甚至是根本性的变化”。为了重现结果，可能需要整个系统的快照。

4.1.6 软件工程过程与管理

ML 和非 ML 系统在开发过程中遵循的过程不同。正如 P2 所建议的，在环境理解的步骤中，与其他利益相关者就机器学习是什么和不能是什么进行沟通是很重要的。P6 提到，一些利益相关者可能会误解机器学习的能力：这些利益相关者通常高估了机器学习技术的效果。机器学习不是灵丹妙药；其效果在很大程度上取决于他们所拥有的数据【✓ S21】。

P14 提到数据处理对整个过程的成功至关重要【✓ S22】，“垃圾进垃圾出，值得花时间”。P12 指出，数据可视化在理解特征分布和相关性以及识别数据中有趣的趋势和模式方面起着至关重要的作用。正如 P6 所指出的，

领域知识是理解数据特性的关键：然而，有时候领域专家不愿意分享他们的知识。他们可能害怕被自动化软件取代，或者没有任何合理的理由只凭直觉。

在一次测试中很难开发出一个好的模型 (P6)。这个步骤的目的是通过尝试和错误找到正确的平衡。即使是最优秀的机器学习实践者也需要对模型进行修修补补。有时候，实践者可能会回到数据步骤来分析错误。

正如 P3、P4 和 P5 所提到的，实践者在生产过程中使用机器学习模型作为端点创建 API 或 Web 应用程序。实践者还需要计划使用更新的数据对模型进行再训练的频率。在监视步骤中，将随时间跟踪模型的性能。一旦数据以使模型失效的方式发生变化，软件就应该准备好对错误和意外结果作出响应。

受访者认为 ML 开发管理与非 ML 开发管理的显著区别在于 ML 开发管理缺乏具体的、实用的指导【S23】。相对于非 ML 软件的开发，ML 系统的开发本质上是一个迭代优化的任务，只要量化的度量满足预期 (P1)，“就有不止一个正确的答案”。有时，可用的数据不足以支持应用程序环境。无论训练的模型有多好，都不可能达到预期的

量化措施。P6 解释道：在我们完成对模型的训练之前，没有人知道量化指标是否可以实现。

受访者还提到，ML 系统的开发计划比非 ML 软件系统更加灵活。模型训练的过程通常不是线性的。

4.2 RQ2. 工作特点差异

在本节中，我们将讨论 ML 和非 ML 开发在工作特性方面的差异。在我们的访谈或焦点小组中，没有从几个工作特性主题中出现共同的主题（工作调度自主性、任务多样性、重要性、来自工作的反馈、信息处理、专门化、来自其他人的反馈、社会支持、工作条件、人体工程学、经验意义、经验责任、结果知识）。因此，我们不在本节中讨论它们。

4.2.1 技能多样性

受访者发现，在技能多样性方面，ML 和非 ML 开发之间存在两个主要差异。

首先，受访者指出，开发机器学习框架和应用程序面临着明显的技术挑战。例如，P10 认为，ML 开发除了需要编程技能外，还需要“数学、信息论和统计方面的专业知识”【✓ S24】。P13 将差异解释为：[对于非 ML 软件系统的开发]开发人员一旦学习了编程语言，就可以为特定的业务需求编写代

码……[对于 ML 系统的开发]数学和统计专业知识是为特定业务需求设计有效模型的重要先决条件。然而，熟练的编程技能仍然很重要，并且可以帮助实现模型。

其次，受访者认为 ML 开发需要更广泛的技能【S25】，如果开发人员缺乏这些技能，那么 ML 开发将更具挑战性。正如 P5 所建议的，“除了编程技能，ML 开发还需要数据分析技能”。P10 总结了数据分析技能包括获取、清理、探索和建模数据的能力。正如 P6 所指出的，ML 开发中的海量数据给数据分析带来了新的挑战：在大数据背景下，仅仅进行统计数据分析是不够的。开发人员应该能够处理大量数据的数据分析。例如，开发人员需要编写用于数据分析的分布式程序和使用分布式计算框架的技能。

4.2.2 工作的复杂性和问题的解决

访谈显示，ML 开发和非 ML 开发在不同方面呈现出复杂性。正如 P12 所示，非 ML 开发的工作复杂性在于架构设计和模块划分【✓ S26】；相反，ML 开发的工作复杂性在于数据建模【S27】。P14 解释说：“不同机器学习应用程序的体系结构是相对固定的，它们通常由几个模块组成，即、数据收

集、数据预处理、数据建模和测试”。

工作复杂性的差异导致解决问题的差异。受访者提到,对于非 ML 开发,通常存在一个清晰的路线图来产生一个好的架构设计和模块划分【✓ S28】。然后,开发人员可以采用逐步实现每个模块的方法(P5、P6、P8)。然而,对于 ML 开发,没有明确的路线图来构建有效的数据模型(P2)。如 P6 所示,与非 ML 开发相比,ML 开发中的问题求解过程具有更多的不确定性:我们不知道数据能告诉我们什么结果,数据模型能在多大程度上得到改进。我们会尝试任何我们认为可能奏效的方法。因此,搜索空间会变得相当大,工作负载可能会相应地激增。有时候,更多的工作负载确实会带来更好的性能。

4.2.3 任务同一性

受访者表示,在任务认同方面几乎没有差异。P4 和 P5 提出了一个不同之处,他们报告说在 ML 开发中很难对任务做出准确的计划【✓ S29】。P4 将原因总结为:在非 ML 软件开发中,可以根据功能点将项目划分为不同的任务。开发人员可以很容易地判断完成特定功能点的实现需要多长时间。然而,在机器学习开发中,数据建模是一个不可分割的任务。为了达到可接受的性能,搜索空间通常相当大。为这样

一项任务制定准确的计划是困难的。

表格 3 克利夫误差值释义

Cliff's Delta Value	Interpretation
$ \delta < 0.147$	Negligible
$0.147 \leq \delta < 0.330$	Small
$0.330 \leq \delta < 0.474$	Medium
$ \delta \geq 0.474$	Large

此外,受访者指出,ML 开发人员对他们的任务进展到目标性能的控制较少(P9, P10)【S30】。一旦开始在 ML 开发中进行数据建模,艰苦的工作可能并不总是能够带来令人满意的结果(P4)。

4.2.4 组织外部的互动

受访者表示,ML 开发人员在与客户沟通时面临更多的挑战【✓ S31】。P2 解释说:由于数据建模的非线性过程,使得从机器学习开发到项目进度的沟通变得更加困难。机器学习开发的结果并不容易解释。例如,很难解释为什么神经网络适用于图像处理。

4.3 调查结果

我们将调查结果汇总在表 2 中。报表列显示了呈现给受访者的报表。下一列显示了我们在整个论文中用来标识语句的标签。李克特分布的四个子列显示了每一组受访者(ML 实践者-ML, 非 ML 实践者-非 ML, ML 框架/工具/库实践者-ML FTL, ML 应用程序实践者-ML App)的协议分布。对于李克特分布,最左边的条表示强烈的不同意,

中间的条表示中立，最右边的条表示最强烈的同意。例如，大多数机器学习实践者强烈同意 S24。

P-value 列表示第 1 个子列中 ML 与非 ML、第 2 个子列中 ML FTL 与 ML App 对每个陈述的一致性是否有统计学意义。表按 p 值排序，在第一个子列中使用 benjamin - hochberg 校正。95%置信水平 (benjamin - hochberg 校正后的 p 值 < 0.05) 差异有统计学意义，用绿色表示。

效应量列表示第一个子列中 ML 和 non-ML 之间的差异，以及这些条件子列中 MLFTL 和 ML App 之间的差异。我们使用 Cliff 的增量来测量差异的大小，因为 Cliff 的增量据说比 Cohen 的 [32] 更健壮可靠。Cliff ' 增量表示两个样本分布之间的重叠程度，范围从 -1 到 +1。极值 ±1 发生在两组交集为空集时，当比较组趋于重叠时，Cliff ' 增量趋于零。效应大小另外颜色从蓝色，橙色基于梯度差异较大的解释指悬崖的三角洲表 3: 蓝色意味着前者更有可能同意的声明，和橙色意味着后者更有可能同意声明。

总的来说，调查结果证实了受访者的说法存在一些差异。根据观察到的 ML 和非 ML 开发在统计学上的显著差异，我们可以肯定地说：

- **需求:** 在 ML 系统开发过程中收集需求需要进行更多的初步实验，这就需要性能的可预测下降。【✓ S2, ✓ S3】

- **设计:** ML 系统的详细设计更耗时，而且往往以密集迭代的方式进行。【✓ S7】

- **测试和质量:** 为 ML 开发收集测试数据集需要更多的工作；良好的测试性能并不能保证 ML 系统在生产中的性能。【✓ S15, ✓ S17】

- **过程和管理:** 数据的可用性通常限制 ML 系统的能力；数据处理对于整个过程的成功往往更为重要。【✓ S21, ✓ S22】

- **技能多样性:** ML 开发需要大量的数学、信息论和统计学知识。【✓ S24】

- **工作复杂性和问题解决:** ML 实践者对于构建系统有一个不太清晰的路线图。【✓ S28】

- **任务标识:** 为 ML 开发的任务制定准确的计划要困难得多。【✓ S29】

- **互动:** ML 从业者与客户沟通的频率往往较低。【✓ S31】

调查结果并不能证实一些受访者关于 ML 和非 ML 之间差异的说法。例如，ML 系统开发和非 ML 软件开发之间的需求是确定的【S1】。一种解释是，尽管 ML 开发人员交付的模型的某些方面存在不确定性，但是需求本身是确定的。

5. 讨论

5.1 影响

包含不确定性：正如我们的受访者所提到的，不确定性存在于 ML 系统开发的各个方面。

表格 2 调查结果。橙色单元格表示前一组（ML 与 ML FTL 从业人员）与后一组（非 ML 与 ML APP 从业人员）相比更不同意该声明；蓝色的单元格表示前一组人更同意的地方。绿色的单元格表示统计中有显著差异的地方。“()” 中数字表示每个组的大小。

		Likert Distributions				Cliff's Delta		P-values	
Statement		ML (98)	Non-ML (244)	ML FTL (39)	ML App (59)	ML	ML FTL	ML	ML FTL
						vs.	vs.	vs.	vs.
						Non-ML	ML App	Non-ML	ML App
Developing my software requires knowledge in math, information theory and statistics.	S24	■■■■■	■■■■■	■■■■■	■■■■■	0.45	-0.19	✓.000	.320
Detailed design is time-consuming and conducted in an iterative way.	S7	■■■■■	■■■■■	■■■■■	■■■■■	0.32	0.18	✓.000	.271
Requirements should consider predictable degradation in the performance of software.	S3	■■■■■	■■■■■	■■■■■	■■■■■	0.29	0.11	✓.000	.433
It is easy to make an accurate plan for the development tasks of my software.	S29	■■■■■	■■■■■	■■■■■	■■■■■	-0.32	0.03	✓.000	.779
Data processing is important to the success of the whole development process.	S22	■■■■■	■■■■■	■■■■■	■■■■■	0.26	-0.20	✓.000	.271
Collecting testing dataset is labor intensive.	S15	■■■■■	■■■■■	■■■■■	■■■■■	0.27	-0.26	✓.000	.188
Developing my software requires frequent communications with the clients.	S31	■■■■■	■■■■■	■■■■■	■■■■■	-0.29	-0.14	✓.000	.577
My software is tested by using automated testing tools.	S18	■■■■■	■■■■■	■■■■■	■■■■■	0.26	0.48	✓.000	.482
Good testing results can guarantee the performance of my software in production.	S17	■■■■■	■■■■■	■■■■■	■■■■■	-0.23	0.09	✓.001	.482
Available data limit the capability my software.	S21	■■■■■	■■■■■	■■■■■	■■■■■	0.22	-0.48	✓.001	.001
Collecting requirements involve a large number of preliminary experiments.	S2	■■■■■	■■■■■	■■■■■	■■■■■	0.20	0.09	✓.002	.577
A clear roadmap exists to build my software.	S28	■■■■■	■■■■■	■■■■■	■■■■■	-0.24	0.07	✓.002	.661
High level architectural design is relatively fixed.	S4	■■■■■	■■■■■	■■■■■	■■■■■	-0.20	0.07	✓.017	.719
Creativity is important during debugging.	S11	■■■■■	■■■■■	■■■■■	■■■■■	0.12	0.07	✓.064	.719
My team puts a lot of effort into maintenance of my software.	S19	■■■■■	■■■■■	■■■■■	■■■■■	-0.15	0.21	✓.065	.719
The higher the performance measures are, the better my software is.	S16	■■■■■	■■■■■	■■■■■	■■■■■	0.08	0.19	✓.068	.719
Architecture design is complicated for my software.	S26	■■■■■	■■■■■	■■■■■	■■■■■	0.10	0.32	✓.069	.047
Data modeling is complicated for my software.	S27	■■■■■	■■■■■	■■■■■	■■■■■	0.07	-0.15	✓.077	.471
Detailed design is flexible.	S6	■■■■■	■■■■■	■■■■■	■■■■■	0.08	0.11	✓.107	.459
Requirements of my software are uncertain.	S1	■■■■■	■■■■■	■■■■■	■■■■■	0.10	0.11	✓.151	.482
Testing involves multiple runs of my software to gather a population of quantitative measures.	S14	■■■■■	■■■■■	■■■■■	■■■■■	0.07	0.06	✓.152	.665
My coding workload is heavy.	S8	■■■■■	■■■■■	■■■■■	■■■■■	0.07	0.08	✓.223	.682
I have control over the progress towards the target performance.	S30	■■■■■	■■■■■	■■■■■	■■■■■	0.06	0.02	✓.265	.756
Code reuse happens frequently across different projects.	S9	■■■■■	■■■■■	■■■■■	■■■■■	0.06	-0.12	✓.265	.482
Debugging aims to locate and fix bugs in my software.	S10	■■■■■	■■■■■	■■■■■	■■■■■	0.06	-0.01	✓.358	.943
Creating my software requires a team of people, each with different skills.	S25	■■■■■	■■■■■	■■■■■	■■■■■	0.04	0.09	✓.540	.482
Testing results of my software are hard to reproduce.	S13	■■■■■	■■■■■	■■■■■	■■■■■	0.04	0.04	✓.546	.787
Low coupling in the components of my software is important.	S5	■■■■■	■■■■■	■■■■■	■■■■■	-0.01	0.08	✓.861	.482
Configuration management are mainly for the code.	S20	■■■■■	■■■■■	■■■■■	■■■■■	-0.07	-0.14	✓.894	.943
Software engineering management lacks practical guidance for my software.	S23	■■■■■	■■■■■	■■■■■	■■■■■	-0.01	-0.20	✓.894	.482
Bugs in my software usually reside in the code.	S12	■■■■■	■■■■■	■■■■■	■■■■■	-0.01	-0.05	✓.979	.787

首先，不确定性来自作为需求的数据。虽然 ML 系统的开发团队有一个目标要达到，例如建立一个绝对精确的语音识别软件，但是为了确保这个目标是可以实现的，需要大量的前期实验，并且有足够的数据来满足这个目标。在需求收集和分析阶段，理解应用程序环境、快速数据可视化和在小规模数据集上动手进行实验数据建模有助于加快初步实验的进度。对于机器学习实践者来说，与其尝试许多工具，不如专注于几个学习和使用[22]的工具。ML 开发中前期实验的探索过程类

似于科学编程[9]，可以借鉴科学编程的经验。

第二，不确定性源于机器学习算法固有的随机性。机器学习实践者应该转变他们的思维方式，接受不确定性。例如，机器学习算法可以初始化为

随机状态；随机噪声有助于在梯度下降(随机梯度下降)过程中有效地找到最优解。为了减少不确定性，机器学习实践者可以通过使用相同的代码、数据和初始随机状态，在一定程度上实现可重复性。因此，代码、数据和参数的版本控制工具链对于实现可重现性[5]是必不可少的。然而，存储所有状态可能会带来巨大的开销，并减慢开发过程。因此，此类工具链的有效性有待于未来的研究。此外，为了评估机器学习算法的性能，从业者通常将数据随机分割成训练和测试集，或者使用 k-fold 交叉验证。机器学习算法的性

能应该报告为度量值的分布，而不是我们的参与者所强调的单个值。

数据处理：正如我们的受访者所讨论的，数据在 ML 系统的开发中扮演着重要的角色。大量的训练数据会对 ML 系统的性能产生巨大的影响。正如与会者所证实的，数据收集成为 ML 系统开发的关键挑战之一。数据收集文献主要关注三方面的研究：发现、扩充或生成数据集的数据采集技术、标记单个数据点的数据标记技术，以及传输学习技术以改进现有数据集。未来的研究可以将现有的数据收集技术集成到 ML 开发过程中。此外，现有的数据收集技术往往是特定于应用程序或数据的。将这些建议的技术推广到各种应用程序需要付出更多的努力。ML 实践者还使用分布式平台并行处理大量数据。调试用于数据建模的并行计算既耗时又容易出错。正如[16]所示，未来的研究可以更加努力地促进 ML 系统的交互和实时调试。

除了数据量之外，质量也是构建强大而健壮的 ML 系统的关键。“垃圾输入，垃圾输出”，实践者从机器学习软件中获得的是他们输入到软件中的内容的表示。真实世界的的数据由缺失的值、不平衡的数据、异常值等组成。在构建模型之前，机器学习实践者必

须处理数据。未来的研究可以开发出数据可视化工具，对数据进行概述，帮助定位不规范之处，使从业者能够专注于数据实际需要清理的地方。然而，在开发过程中，高质量的数据集并不能永远保证机器学习系统的高性能。在一个快速发展的环境中，机器学习系统一旦投入生产[34]，其精度就会下降。实践者需要认识到，机器学习系统从来没有最终版本，它需要随着时间不断更新和改进(例如，提供新的数据和重新培训模型)。在线反馈和 ML 系统的性能度量是未来研究的沃土。

5.2 有效性威胁

内部效度：有可能我们的调查对象中有一些人对评分的陈述理解较差。他们的反应可能会给我们收集的数据带来噪音。为了减少这个问题的影响，我们在调查中加入“我不知道”选项，并忽略了标记为“我不知道”的回答。我们还删除了那些工作角色不属于这些人提交的响应：软件开发、测试和项目管理。两位作者将我们的调查翻译成中文，以确保来自中国的受访者能够很好地理解我们的调查。为了减少用双语表述调查结果的偏差，我们仔细地翻译了调查结果，以确保英语和汉语术语之间没有歧义。我们还根据试点调查的反馈，通过提高清晰度

和可理解性来完善翻译。

在这项研究中，ML 和非 ML 开发之间有统计学上显著差异的效应大小从微不足道到中等不等。可忽略的效应大小表明，机器学习和非机器学习开发之间的特殊差异是微不足道的，即使它在统计上是显著的。为了减轻这一威胁，我们在结果中没有强调这些差异。

作为我们的调查对象，我们邀请各种可能涉及 ML 生态系统的不同部分 (ML 框架、工具、库、带有 ML 组件的软件应用程序) 的潜在受访者。当我们研究 ML 和非 ML 发展之间的差异时，我们混合了这些 ML 应答者的回答。这两组人的认知可能存在差异，并压倒了 ML 和非 ML 的差异。为了防止这种威胁，我们比较了两组之间李克特反应分布的差异。

为了从 ML 和非 ML 人群中招募受访者，我们将调查范围广泛地扩展到来自世界各地的公司。在调查开始时，我们明确表示，我们研究的目的是了解机械盈利是否以及如何改变软件工程实践。这种描述可能会吸引一部分了解 ML 的非 ML 人群的更多关注，但是 ML 不是他们日常工作的一部分。此外，描述可能会产生一个默认的假设，即机器学习改变软件工程实践。这种

假设可能会误导受访者夸大他们所感知到的差异。

外部效度：为提高我们的研究结果的普遍性，我们采访了来自三家公司 14 个受访者，调查了来自 26 个国家的 342 名受访者，他们为不同的公司 (如亚马逊、阿里巴巴、百度、谷歌、恒田、IBM、英特尔、IGS、柯达、联想、微软、摩根士丹利) 工作，或在 GitHub 上以不同的角色为开源机器学习项目做贡献。

不过，我们希望强调的是，虽然我们从三家中国 IT 公司中挑选了员工进行面试，但通过更多 IT 公司参与的焦点小组讨论，我们提升了来自面试的反馈，而且调查对象相当广泛。从访谈中得到的改进的回答被用来引导我们在调查中对这些陈述进行评分。调查容许受访者在适当时透过自由表格填写新增意见。查看这些字段中的回复时，我们没有发现任何缺少语句的迹象。

此外，我们的受访者报告的一些说法没有通过调查得到证实，而且这些说法可能还为时过早。

6. 相关工作

一些先前的工作为机器学习系统的开发提供了规范的实践 (例如，[29])。一些人讨论了该行业的现实挑

战和最佳实践，例如亚马逊[33]的机器学习模型管理和谷歌[40]的机器学习工程最佳实践。研究了基于栈溢出[38]的机器学习相关问题。这些工作基于作者的经验，并且在很大程度上没有将机器学习开发作为一种特殊类型的软件工程进行环境。相反，我们的发现是基于经验观察，明确地关注 ML 和非 ML 开发之间的差异。

和我们的工作一样，一些研究人员也对数据科学的软件工程进行了实证研究。一些人关注在公司内部工作的数据科学家如何通过访谈，从一般工具[13]的角度识别痛点，并探索采用可视化分析工具[20]的挑战和障碍。其他关注于描述与数据科学相关的专业角色和实践：Harris 等人[17]调查了 250 多名数据科学从业者，以对数据科学从业者进行分类并确定他们的技能集。Kim 等人采访了 16 位微软的数据科学家，确定了[21] 5 种工作方式，并补充了 Harris 等人对[22]不同活动的工具使用、挑战、最佳实践和时间的调查。与之前的工作相比，我们的论文研究了 ML 和非 ML 开发之间的广泛差异。

与我们的研究最相似的是对微软团队[4]将人工智能功能集成到软件和服务以及最佳实践中的实证研究。

依据微软团队提出的 ML workflows，他们确定了 11 个挑战，这些挑战包括 1) 数据可用性、收集、清洗、和管理，2) 教育和培训，3) 硬件资源，4) 端到端的管道支持，5) 协作和工作文化，6) 规范，7) 将人工智能集成到更大的系统，8) 指导和辅导，9) 人工智能工具，10) 规模、进化和 11) 模型，评估，和部署。已确定的挑战出现在不同的软件开发实践中。我们的研究结果在以下几个方面与他们的不同：

•**设计：**两项研究都认为在 ML 开发中维护模块化是困难的。他们的研究[4]将其原因归结为 ML 模型的可扩展性较低以及 ML 模型之间的交互作用不明显。相反，我们发现 ML 开发与非 ML 开发一样重视组件中的低耦合【S5】。两项研究都认为，在 ML 系统的详细设计中存在快速迭代【✓ S7】。

•**建设和工具：**两项研究都认为，由于应用程序环境和输入数据的不同，ML 开发中的代码重用具有挑战性。尽管在 ML 开发中存在代码重用的挑战，但是我们发现代码重用在 ML 开发中与在非 ML 开发中一样频繁【S9】。

•**流程和管理：**两项研究都认为，由于数据的参与，ML 开发的管理具有挑战性，而数据的可用性通常限制 ML 系统的能力【✓ ✓ S21】。

•**配置管理**：两项研究都认为在 ML 开发中需要数据版本控制。尽管有必要进行数据版本控制，但是我们发现 ML 开发中当前的配置管理活动仍然关注代码版本控制【S20】。除了数据版本控制，早期的研究[4]建议跟踪数据是如何收集和处理的。

如上所述，我们的研究证实了 Amershi 等人工作中报道的一些发现。与 Amershi 等人的工作不同，由于我们在访谈中遵循了 SWEBOK，并考虑了应用心理学领域的工作特点，所以我们认识到了 ML 和非 ML 开发在其他方面的差异，例如需求收集、工作复杂性、问题解决过程和任务识别。此外，我们还从更广泛的人群中收集了意见，例如，包括来自不同软件行业的开源从业者和专业人士。

7. 结论

在这项工作中，我们确定了 ML 和非 ML 开发之间的显著差异。区别在于不同的方面，包括软件工程实践(例如，ML 开发中的探索性需求引出和迭代过程)和软件开发环境(例如，ML 开发中的高复杂性和对独特解决方案和需求的需求)。这种差异源于机器学习的固有特征——不确定性和所使用的数据。

为了解决不确定性，ML 实践者应该转变他们的思维，在初步实验中接

受不确定性和 ML 算法的随机性。他们可以从科学规划中吸取教训，科学规划也包括开发过程中的探索过程。此外，用于代码、数据和参数的版本控制工具链对于 ML 实践者实现可重现性起着至关重要的作用。ML 实践者还应该投入足够的精力来处理要使用的数据。未来的研究可以更加努力地提供交互式 and 实时的调试工具，以促进 ML 系统的有效开发。为了应对数据的快速发展，ML 系统的在线反馈和性能度量是未来研究的沃土。

从更广泛的意义上说，这项工作代表了理解软件开发的一个步骤，它不是一个同质体，而是一个丰富多彩的实践织锦，其中包含了来自不同领域的不同背景的人。精确的差别可能存在于不同类型的 ML 架构中。我们把这些问题的研究留给未来的研究。

感谢

作者感谢受访者的参与，也感谢受访者对我们的调查做出的回应。

文献参考

- [1] Nvivo qualitative data analysis software, 2018.
- [2] The amazon machine learning process. <https://docs.aws.amazon.com/machine-learning/latest/dg/themachine-learning-process.html>, 2019. Mar. 2019.
- [3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, pages 265 – 283, 2016.
- [4] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software engineering for machine learning: A case study. In Proceedings of the 39th International Conference on Software Engineering – SEIP track. IEEE Computer Society, May 2019.
- [5] A. Anjos, M. Günther, T. de Freitas Pereira, P. Korshunov, A. Mohammadi, and S. Marcel. Continuously reproducing toolchains in pattern recognition and machine learning experiments. In Proceedings of the 34th International Conference on Machine Learning, Aug. 2017. <https://openreview.net/group?id=ICML.cc/2017/RML>.
- [6] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, et al. Tfx: A tensor flow-based production-scale machine learning platform. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1387 – 1395. ACM, 2017.
- [7] P. Bourque, R. E. Fairley, et al. Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press, 2014.
- [8] V. Braun and V. Clarke. Using thematic analysis in psychology. Qualitative research in psychology, 3(2):77 – 101, 2006.
- [9] J. C. Carver, R. P. Kendall, S. E. Squires, and D. E. Post. Software development environments for scientific and engineering software: A series of case studies. In Proceedings of the 29th International Conference on Software Engineering, pages 550 – 559. IEEE Computer Society, 2007.
- [10] J. Cohen. A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1):37 – 46, 1960.
- [11] G. Ericson. What is the team data science process? <https://docs.microsoft.com/en->

- us/azure/machinelearning/team-data-science-process/overview, 2017. Mar. 2019.
- [12] A. Ferlitsch. Making the machine: the machine learning lifecycle. <https://cloud.google.com/blog/products/ai-machinelearning/making-the-machine-the-machine-learning-lifecycle>, 2019. Mar. 2019.
- [13] D. Fisher, R. DeLine, M. Czerwinski, and S. Drucker. Interactions with big data analytics. *interactions*, 19(3):50–59, May 2012.
- [14] Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.
- [15] G. Guest, A. Bunce, and L. Johnson. How many interviews are enough? an experiment with data saturation and variability. *Field methods*, 18(1):59–82, 2006.
- [16] M. A. Gulzar, M. Interlandi, S. Yoo, S. D. Tetali, T. Condie, T. Millstein, and M. Kim. Bigdebug: Debugging primitives for interactive big data processing in spark. In *Proceedings of the IEEE/ACM 38th International Conference on Software Engineering*, pages 784–795. IEEE, 2016.
- [17] H. Harris, S. Murphy, and M. Vaisman. Analyzing the Analyzers: An Introspective Survey of Data Scientists and Their Work. ” O’ Reilly Media, Inc.”, 2013.
- [18] S. E. Humphrey, J. D. Nahrgang, and F. P. Morgeson. Integrating motivational, social, and contextual work design features: a metaanalytic summary and theoretical extension of the work design literature. *Journal of applied psychology*, 92(5):1332, 2007.
- [19] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.
- [20] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, 2012.
- [21] M. Kim, T. Zimmermann, R. DeLine, and A. Begel. The emerging role of data scientists on software development teams. In *Proceedings of the 38th International Conference on Software Engineering*, pages 96–107. ACM, 2016.
- [22] M. Kim, T. Zimmermann, R. DeLine, and A. Begel. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering*, 44(11):1024–1038, 2018.
- [23] B. A. Kitchenham and S. L. Pfleeger. Personal opinion surveys. In *Guide to advanced empirical software engineering*, pages 63–92. Springer, 2008.
- [24] T. D. LaToza, G. Venolia, and R. DeLine.

- Maintaining mental models: A study of developer work habits. In Proceedings of the 28th International Conference on Software Engineering, ICSE '06, pages 492–501, New York, NY, USA, 2006. ACM.
- [25] K.-c. Lee, B. Orten, A. Dasdan, and W. Li. Estimating conversion rate in display advertising from past performance data. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 768–776. ACM, 2012.
- [26] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, pages 120–131. ACM, 2018.
- [27] S. Ma, Y. Aafer, Z. Xu, W.-C. Lee, J. Zhai, Y. Liu, and X. Zhang. Lamp: data provenance for graph based machine learning algorithms through derivative computation. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pages 786–797. ACM, 2017.
- [28] S. Ma, Y. Liu, W.-C. Lee, X. Zhang, and A. Grama. Mode: automated neural network model debugging via state differential analysis and input selection. In Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pages 175–186. ACM, 2018.
- [29] A. Ng. Machine learning yearning. URL: <http://www.mlyearning.org>, 2017.
- [30] K. Pei, Y. Cao, J. Yang, and S. Jana. Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles, pages 1–18. ACM, 2017.
- [31] Y. Roh, G. Heo, and S. E. Whang. A survey on data collection for machine learning: a big data-ai integration perspective. arXiv preprint arXiv:1811.03402, 2018.
- [32] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek. Appropriate statistics for ordinal level data: Should we really be using t-test and cohensd for evaluating group differences on the nsse and other surveys. In Proceedings of the Annual Meeting of the Florida Association of Institutional Research, pages 1–33, 2006.
- [33] S. Schelter, F. Biessmann, T. Januschowski, D. Salinas, S. Seufert, G. Szarvas, et al. On challenges in machine learning model management. <http://sites.computer.org/debull/A18dec/p5>.

pdf, 2018. Mar. 2019.

[34] D. Talby. Lessons learned turning machine learning models into real products and services.

<https://www.oreilly.com/ideas/lessons-learned-turning-machine-learning-models-into-real-products-and-services>, 2018. Mar. 2019.

[35] R. Thomas. What do machine learning practitioners actually do?

<https://www.fast.ai/2018/07/12/auto-ml-1/>, 2018. Mar. 2019.

[36] Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the 40th international conference on software engineering, pages 303 – 314. ACM, 2018.

[37] P. K. Tyagi. The effects of appeals, anonymity, and feedback on mail survey response patterns from salespeople. Journal of the Academy of Marketing Science, 17(3):235 – 241, Jun 1989.

[38] Z. Wan, J. Tao, J. Liang, Z. Cai, C. Chang, L. Qiao, and Q. Zhou. Large-scale empirical study on machine learning related questions on stack overflow. Journal of Zhejiang University (Engineering Science), 53(5):819 – 828, 2019.

[39] X. Xie, J. W. Ho, C. Murphy, G. Kaiser,

B. Xu, and T. Y. Chen. Testing and validating machine learning classifiers by metamorphic testing. Journal of Systems and Software, 84(4):544 – 558, 2011.

[40] M. Zinkevich. Rules of machine learning: Best practices for ml engineering.

<https://developers.google.com/machinelearning/guides/rules-of-ml/>, 2018. Mar. 2019.

i