Safe-DS：让数据科学更安全的领域特定语言

Lars Reimann 计算机科学研究所 III 波恩大学，德国 邮箱：reimann@cs.uni-bonn.de

G¨unter Kniesel-W¨unsche 计算机科学研究所 III 波恩大学，德国 邮箱：gk@cs.uni-bonn.de

arXiv:2302.14548v2 [cs.SE] 7 Apr 2023

摘要——由于数据科学（DS）流水线运行时间很长，即使是小小的编程错误，如果未能通过静态方式检测出来，也可能造成高昂代价。然而，由于大多数数据科学流水线都是用 Python 编写的，因此即使是对数据科学流水线进行基础的静态类型检查也很困难。在 Python 中，只有通过外部代码检查器才能实现静态类型检查。这些工具要求为函数的参数或结果提供静态类型注解，而许多数据科学库并未提供这些注解。

本文展示了如何通过 Safe-DS 使用丰富的 Python 数据科学库，以静态安全的方式进行数据科学（DS）开发。Safe-DS 不仅能捕获常规类型错误，还能检测与范围限制、数据操作和函数调用顺序相关的错误，其能力远超当前的 Python 代码检查工具。Python 库通过存根语言集成到 Safe-DS 中，以此指定其声明的接口；此外，*API* 编辑器能够从 Python 库的代码和文档中提取类型信息，并自动生成合适的存根。

此外，Safe-DS 还通过图形表示来补充文本形式的数据科学流水线，从而通过防止语法错误来实现安全开发。文本视图与图形视图的无缝同步使开发者始终可以根据自身技能和当前任务选择最合适的视图。

我们认为，Safe-DS 可以让数据科学开发变得更简单、更快速、更可靠，并显著降低开发成本。

索引词—数据科学、机器学习、静态安全性、精化类型、模式类型、领域特定语言

I. 引言

许多研究人员指出，广泛使用的库的 API 往往难以学习，也很难正确使用 [1]–[4]。在数据科学（DS）中，防止错误使用尤为重要：如果程序在运行时崩溃，而问题本可以在编译时检测到，那么通常会浪费时间、资源，最终还会浪费金钱；更不用说程序运行数小时甚至数天后才崩溃，更是令人无法接受。因此，静态安全性将使每一位数据科学家，无论新手还是专家，都受益匪浅。遗憾的是，大多数数据科学流水线都是用 Python 编写的，并使用 Python 数据科学库 [5]。对于 Python，只有通过 mypy[1]，这类外部代码检查器才能进行静态类型检查；然而，如果代码没有添加类型提示 [6]，这些工具就无法使用。遗憾的是，许多库并未提供类型提示，而且据我们所知，没有任何 Python 代码检查工具能够处理高级类型概念，例如模式类型对于表格数据集而言，这类功能将对典型的数据科学应用大有帮助。

另一种截然不同的方法是使用可视化流水线规范语言（例如 RapidMiner Studio [7]）、向导式界面（例如 RapidMiner Go [8]）或配置语言（例如 Ludwig[9]），将数据科学 API 隐藏起来。然而，向导和配置语言只能通过限制可表达的内容来提供引导 [10] 和安全性。可视化编程可以提供完整的表达能力，但当流水线变得复杂时，图表示很快就会变得杂乱，而文本语言则更具优势。根据用户的技能和使用场景，不同的表示方式各有其最佳适用情况。然而，目前尚无已知系统能够以可视和文本两种方式来规范数据科学流水线。

在此背景下，我们探讨以下研究问题：

- 研究问题 1：如何让编程初学者也能使用最广泛应用的数据科学库?
- 研究问题 2：能否设计一种表达能力足够强、能够实现任意数据科学流水线，同时又足够简单、便于学习和正确使用，甚至让初学者也能掌握的语言?
- 研究问题 3：哪些类别的错误无法被传统类型系统（尤其是 Python 代码检查工具所采用的类型系统）捕获，却在数据科学开发中频繁出现?
- 研究问题 4：哪些类型概念有助于设计专用的数据科学语言，又应纳入其中?
- 研究问题 5：数据科学语言如何确保以静态方式安全地使用以动态语言编写的数据科学库?
- 研究问题 6：如何让现有 Python 数据科学库易于集成，甚至让第三方也能轻松集成?
- 研究问题 7：如何利用图形化语言内置的错误预防机制，同时不牺牲文本代码的灵活性?

本文提出的解决方案是 Safe-DS，一种用于指定数据科学流水线的静态类型领域专用语言（DSL）[11]。它提供：

1) 提供一种简单机制，以类型安全的方式集成现有的 Python 数据科学库（第 IV 节）。2) 精心设计的标准库，包含目前可用的最佳数据科学库（第 VI 节简要介绍）。3) 支持第三方开发者通过集成自己的功能来扩展标准库（第 IV-C 节）。4) 在图形化表示与文本表示之间实现无缝的双向

[1]https://github.com/python/mypy

发表于第45届国际软件工程会议 NIER 专题 ICSE-NIER'23，2023年5月14日至20日，澳大利亚墨尔本 ©2023 IEEE

转换；两者只是同一抽象——领域专用语言（第 V 节）的不同视图。

II. 现有技术

根据2021年 Kaggle 数据科学调查 [5]，，在近26,000名受访者中，超过84%的人经常使用 Python，使其成为最常用的数据科学编程语言。根据该调查，数据科学最常用的库包括用于数据可视化的 Matplotlib [12]，Seaborn [13]，和 Plotly [14]、用于数据处理的 Pandas [15]，以及用于机器学习的 scikit-learn [16], [17], TensorFlow [18], Keras [19], PyTorch [20],和 XGBoost [21] [2]。这些库均使用 Python 实现，凸显了 Python 在数据科学开发中的重要作用。

尽管如此，除 Python 之外，数据科学流水线开发还有一些值得关注的方案：Ludwig [9] 允许用户通过 YAML 配置文件构建数据科学流水线。用户必须指定输入和输出功能，也可以选择修改流水线的某些部分，例如数据预处理。

外部领域特定语言是另一种解决方案。用于数据科学的领域专用语言列表在 2016 [22] 中包括用于数据处理的文本领域特定语言，如 Pig Latin [23]，以及用于表示概率模型的可视化领域特定语言，如 [24]。此后又出现了其他工具，例如 BiDaML [25], [26],，它为数据科学流水线开发的不同阶段提供了五种图示符号。另一个流行工具是 RapidMiner Studio（原名 YALE）[7],，它提供了一种可视化领域特定语言，可快速且安全地构建数据科学流水线。

最后，一些工具会使用向导逐步引导用户完成数据科学流水线的开发，例如 RapidMiner Go [8]。为克服向导灵活性有限的问题，用户可以将最终结果导出到 RapidMinerStudio。

[27] 通过额外的图形用户界面元素（如用于检查数据的电子表格视图）扩展了 Jupyter Notebook [28]。当用户在这些元素中操作数据时，相应的代码会添加到笔记本中。然而，对生成代码所做的更改通常无法反映到图形用户界面中，因此图形视图与文本视图之间的转换只能是单向的。

这些理念可以与自动化机器学习 [29] 结合，根据初始数据自动推导出数据科学流水线的部分内容。

关于数据挖掘免费工具的概览可参见 [30]，而关于数据科学领域的更全面综述可参见[31]。即使将这一更大的背景考虑在内，现有工具也没有一个能为我们大多数研究问题提供全面的解决方案，更不用说解决所有问题了。

III. 数据科学流水线语言

数据科学流水线是执行数据加载、数据准备、模型训练和超参数优化，以生成可供使用的模型的代码。它保留了从相同数据复现相同模型所需的步骤，但不包含开发人员为决定如何构建流水线而执行的数据分析和模型质量评估步骤。

我们的工作的主要目标是为不同技能水平的流水线开发者提供支持，包括编程和数据科学新手。因此，组装数据科学流水线应该易于学习和使用，但不易被误用。任何解决方案都必须提供一个小巧且一致的 API（以便于学习和正确使用），并且易于扩展（以便充分利用数据科学库的发展）。扩展必须可行且简单，第三方库开发者也不例外。

研究问题 *1*：鉴于主要的数据科学库是使用 Python 实现的，编程初学者不可能使用它们来构建流水线。即使对于经验丰富的程序员而言，缺少供代码检查工具使用的类型注解的 Python 库也很容易被误用 [32]。为了让数量庞大的数据科学库（第 II 节）能够供初学者使用，并防止各个技能水平的用户误用这些库，我们提出通过 Safe-DS（一个用于数据科学的领域专用语言）重新使用这些库，由其完成繁重的工作。

研究问题 *2*：这引出了一个问题：哪些语言特性对于不受限制的流水线开发是必需的，哪些则必须避免，以便初学者更容易学习和使用。由于流水线开发并不涉及算法实现，而主要是对已经实现的库函数进行一系列调用，因此所需的语言特性集合很简单。图 1 展示了使用广受欢迎的“泰坦尼克号”示例创建 Safe-DS 流水线的开头部分。

```
流水线 predictTitanicSurvival{val 泰坦尼克号 =
loadDataset("泰坦尼克号");val survived =泰坦尼克号.
keepColumns("Survived");// ...}
```

图 1. 用于预测泰坦尼克号事故期间生还情况的 Safe-DS 流水线开头部分。

关键字 流水线 是 Safe-DS 程序的入口。示例中的流水线名为 *predictTitanicSur-vival*。流水线包含一系列语句，通常是赋值语句[3]。每个赋值语句的右侧都包含待求值的表达式。表达式可以是字面量（*"Titanic"*）、对变量的引用（*titanic*）或调用。调用会调用全局函数（例如 *loadDataset*），或方法（例如 *keepColumns*）。我们将方法和全局函数统称为处理过程。调用可以包含其他表达式作为参数。

处理过程可以将 Lambda 表达式（匿名的、带参数的语句序列）作为参数，以配置其部分行为。例如，一个通用的 *transform* 函数会更改表格数据集某一列中的值，因此需要知道具体如何更改这些值。这类高阶处理过程主要用于数据处理函数，其他情况下很少需要。

值得注意的是，Safe-DS 中没有条件语句、循环或递归调用。这些都封装在流水线中调用的库函数内（例如超参数搜索）。开发者所作的决策和迭代（反思当前流水线的结果、对其进行修改并重新启动）发生在流水线之外。缺少循环和条件语句使得该

[2]每个组均按使用百分比降序排列 [5]

[3]除了赋值语句外，唯一的其他语句是表达式语句，它会对表达式求值以产生副作用，并忽略结果。

语言易于学习。由于数据在线性地流经程序，它也有助于进行全面的静态检查（第 IV 节）。

IV. 静态检查

在数据科学流水线的开发过程中保障安全，是我们的核心目标之一。为此，静态检测错误，也就是在程序运行前检测错误，至关重要，因为在大型数据集上执行数据科学流水线可能需要数小时甚至数天。要实现静态检查，就需要回答研究问题 3 至研究问题 6。

A. 类型问题

对 [32] 的实证评估表明，Python 代码检查工具的基本类型检查功能已经能够捕获许多常见错误。不过，我们也发现了一些超出传统类型系统范畴的问题，这些问题在数据科学语境中尤为常见或重要（研究问题 3）。

缺少范围限制：我们发现，库的文档经常会规定范围限制，例如，比例必须是介于 0 和 1 之间的数值。

数据集访问失效：处理表格数据集时，若要添加、删除或重命名列，很容易忘记更新仍在访问已更改列的某些操作，因为列名只是字符串，类型系统并不了解它们的特殊语义，也不知道如何推断列的类型。

调用顺序错误：机器学习流程通常必须按特定顺序调用（例如先训练再预测）。这个简单示例可以在 API 层面通过区分类型 未训练模型 和已训练模型来解决，只有后者才具有预测 方法。然而，这会排除对原地修改未训练模型的现有库（例如 scikit-learn）的集成，还需要重新设计库，并导致类型数量膨胀。

B. 类型系统

在识别出具有挑战性的类型问题（RQ3）后，我们现在介绍 Safe-DS 提供的解决方案（RQ4）。

基础类型系统：为涵盖数据科学 API 的基本特性，Safe-DS 类型系统支持类类型、枚举类型、函数类型（包括参数和结果的类型）、联合类型、子类型和泛型。

精化类型：作为表达范围限制的一种方式，Safe-DS 还包含精化类型，它通过某种约束来限制现有类型的合法取值集合。对于具有精化类型的参数，只能传入可由编译器求值为常量的表达式。否则，编译器将无法验证精化类型的约束。

模式推断与检查：为解决数据集访问失效问题（第 IV-A 节），Safe-DS 会检查模式敏感操作的正确性，而无需运行截至该处的整个数据科学流水线：(1) 从外部来源（例如 CSV 文件）加载表格数据集时，编译器会通过分析数据推断其初始模式。(2) 对于流水线中调用的任何数据处理函数，编译器会根据输入数据集的模式以及函数的模式效应描述，归纳推断结果数据集的模式。我们的模式效应系统 [33], [34] 可涵盖列添加、删除、重命名或类型变更的任意组合。此外，我们还支持两种基本的模式检查：(1) 确保存在具有指定名称的列。(2) 确保列具有指定类型。基于模式推断结果，编译器会静态执行此类检查，从而尽早向用户提供反馈。

顺序检查：为了解决顺序问题，我们在接口描述中支持行为协议 [35] 。这些正则表达式描述了合法的调用顺序，其中基本标记不是字符，而是处理过程。例如，我们可以指定，对于任何模型，都必须先调用训练模型的处理过程，再调用进行预测的处理过程。利用这些信息，我们可以静态地确保流水线中的调用顺序正确。

C. 集成外部代码

到目前为止，我们尚未解释 Safe-DS 编译器如何得知有哪些函数可用，以及如何静态检查以不提供静态类型系统的语言实现的函数调用（RQ 5）。

将一组固定的处理过程硬编码到 Safe-DS 的语法中，显然会妨碍可扩展性。我们可以通过分析外部库的代码，收集其中包含的类型提示，或者在用户开发流水线时推断参数和结果的类型，来避免这一问题。然而，许多 Python 数据科学库不提供类型提示，而且通常无法对 Python 进行类型推断。

因此，我们将 Safe-DS 设计为第 III 节所述的流水线语言 与附加的存根语言的组合。每个存根都准确描述了特定 Python 函数的完整类型化接口。图 2 展示了*loadDataset* 过程的存根。*fun* 关键字表示*loadDataset* 是一个函数。对于每个参数（位于括号内）或结果（位于箭头之后），我们列出其名称和
类型，两者以冒号分隔。

```
函数 loadDataset(name: String)-> 数据集: 数据集
```

图 2. 用于*loadDataset* 函数的 Safe-DS 存根。

流水线现在可以调用指定的*loadDataset* 过程（图 1）。Safe-DS 编译器通过比较推断出的参数类型与相应存根中指定的类型，检查调用的类型正确性。对于任何没有匹配存根的过程调用，系统都会判定其有误并显示错误消息。

存根会告知 Safe-DS 编译器进行类型检查所需的全部信息。因此，存根语言还包含用于指定精化类型、模式效应、模式检查和行为协议的方式。例如，图 3 中的过程 *get column* 会返回给定名称的表中的列，并规定表中必须存在名为*name* 的列。

```
函数getColumn(table: 表, name: 字符串)
->列: 列
其中name 位于table
```

图 3. 带模式检查的函数存根，用于确保 *name* 列存在于表中。

D. 半自动存根生成与实现

借助存根语言，新机器学习模型或其他实用函数的实现者可以将其集成到 Safe-DS 中。然而，手动集成整个库会十分繁琐，并会严重阻碍 Safe-DS 的采用。因此，我们需要说明如何简化这一过程（RQ 6）。

我们采用并改进了 [32]提出的 API 编辑器方案。首先，根据代码中的类型提示推断 Python 代码的类型 [6] ；如果没有类型提示，则根据文档推断。对于无法自动推断的情况，我们提供了一个易于使用的图形用户界面。它还支持指定对 API 的更改（例如，删除声明、重命名以及创建参数对象）。其中一些建议甚至可以自动推断。改进并简化后的 API 最终确定后，系统会自动创建 Safe-DS 存根及其对应的实现，即作为原始库包装器的 Python 包装器。

这样，只需少量人工干预（仅限文档不够准确的情况），即可快速将大型库集成到 Safe-DS 中。

V. 不同视图

到目前为止，我们讨论了如何以文本形式表示 Safe-DS 数据科学流水线（第 III 节）。然而，这种方式只适合具备一定编程技能的人，而参与数据科学流水线开发的人员并非人人都具备这类技能。为了鼓励不具备编程技能的人尝试解决数据科学问题，并理解专家的解决方案，我们针对不同的技能组合提供了数据科学流水线的不同视图（RQ 7）。

图 4 展示了将图 1 中的流水线转换为可视化领域特定语言的一种可能方式。这种语言仍然提供完全的灵活性，同时通过防止语法错误来提高安全性。外围方框表示流水线，右上角显示其名称。内部的圆角矩形表示调用的处理过程，左侧的圆圈表示参数，右侧的圆圈表示结果。这些信息取自处理过程的存根（第 IV-C 节）。

箭头表示处理过程之间的数据流，例如，从*loadDataset* 指向*keepColumns* 处理过程的箭头，表示第一个处理过程的结果是第二个处理过程的输入。箭头上方显示的名称对应于文本领域特定语言中的变量名。最右侧的箭头连接着

连接到一个“空白”圆圈，因为这个值从未被使用，但我们仍想显示存储该值的变量名称，以便更轻松地映射到文本表示，并为可能尚未完成的流水线提供提示。为避免图表示过于杂乱，字面量输入（例如字符串）会被隐藏。点击某个流程后即可使其显示出来。这种做法有助于保持图表简洁。RapidMiner Studio [7] 也采用了这种方式。

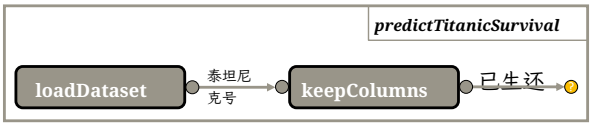

图 4. 图 1 中流水线的图形表示。只有单击节点时才会显示字面量，以保持图形简洁。

现有系统（第 II 节）要么提供流水线的代码视图，要么提供图视图；对于 [27] 两种视图兼有的系统，也只完整定义了从图视图到代码视图的转换。我们认为，支持这两种视图并允许双向无缝转换至关重要，这样参与数据科学流水线开发的每个人都能使用最适合自己的视图，并根据需要在不同视图间切换。例如，这将有助于领域专家与经验丰富的数据科学家 [36], [37]开展协作：领域专家可以使用可视化领域特定语言构建初始流水线原型，然后将其交给数据科学家，由后者使用文本领域特定语言进一步完善和优化。之后，数据科学家可以再次使用可视化领域特定语言向领域专家讲解自己的解决方案，双方还可以一起做出进一步修改。

只有当目标视图的表达能力至少与源视图相当时，才能在不同视图间切换。由于我们希望能够在文本视图和图形视图之间无缝地双向切换，这意味着两种表示必须等价。我们的方法自然保证了这一难以实现的前提。在 Safe-DS 中，文本和图形只是同一 语言的两种不同表示法。

VI. 原型

语言： 我们使用 Xtext[5]实现了 Safe-DS[4]的一个原型。截至撰写本文时，该原型已包含流水线和存根语言的完整文本语法、作用域管理、Safe-DS 代码的自动格式化、Python 代码生成，以及大量静态检查。目前，静态检查支持不含泛型和精化类型的基础类型系统，也支持流水线中的类型推断。模式效应检查仍在开发中，行为协议则是未来的工作。我们使用 Xtext 创建了一个语言服务器 [38], ，并通过一个小型扩展[7]为 Visual Studio Code[6]提供该语言服务器。

元模型：语言的核心是元模型，即对语言元素的抽象表示，也称为语言的抽象语法。各个数据科学流水线都是该元模型的实例。静态检查基于元模型实现。领域专用语言的每种视图（第 V 节）都定义了与元模型之间的转换。首先，这使静态分析能够独立于具体视图定义，

[4]https://github.com/Safe-DS/DSL [5]https://www.eclipse.org/Xtext [6] https://code.visualstudio.com [7] https://marketplace.visualstudio.com/items?itemName=safe-ds.safe-ds

具体视图只需以适当的形式向用户显示错误。其次，我们无需定义并实现当前及未来所有视图之间的转换。该元模型使用 Eclipse建模框架[8]实现。

图5展示了文本视图和图形视图与元模型之间的关系，也展示了从元模型到编译器生成的 Python 代码的单向转换。

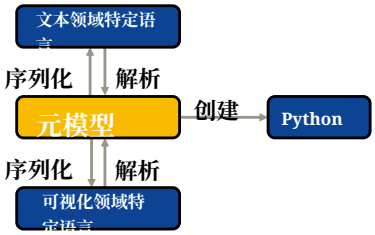

图5. 不同视图（蓝色）之间的交互，以及元模型如何将它们连接起来。

标准库：此外，我们开始着手开发 Safe-DS 标准库。目前，它包含用于处理表格数据（内部使用 pandas [15]）、可视化这些数据（使用 seaborn [13]），以及训练和评估机器学习模型（使用 scikit-learn [16], [17]）的基本功能。相应 Python 实现的实验版本已在 PyPI[9] 上发布。我们将在未来的论文中更详细地讨论该标准库。

VII. 讨论

存根的准确性：静态分析要求存根准确描述相应 Python 实现的前置条件。若两者不匹配，可能导致有效的领域特定语言程序被拒绝，也可能导致无效的领域特定语言程序被执行，从而引发运行时异常或未定义行为。我们旨在尽可能通过自动为 Python 声明创建存根来降低此类不匹配的可能性（第 IV-D 节）。此外，只有在需要集成标准库中缺失的功能时，用户才需要使用存根语言。对于标准库本身，我们尤其重视存根的准确性。

省略语言概念：虽然该语言本身不包含循环或条件语句，但用户仍可在 Python 中实现高阶 *if* 或 *while* 函数，并通过其存根语言将这些函数集成到 Safe-DS 中。我们会像处理其他函数一样处理它们，只使用存根中的信息进行静态分析，或在图形视图中呈现它们。一方面，这意味着我们在静态分析时会忽略 if 和 while 结构的语义。另一方面，这类结构本来就没有必要，因为我们提供了领域特定的抽象，例如用于表格数据集的 *transform* 函数。该函数封装了 Python 实现中的所有循环或条件语句，并利用领域专用语言的功能（例如模式检查，第 IV-B 节）进行领域特定验证。

[8]https://www.eclipse.org/modeling/emf [9]https://pypi.org/project/safe-ds

VIII. 未来计划

其他检查：我们正在努力完成模式检查的实现，之后将使用行为协议添加调用顺序检查。我们还在研究是否可以扩展 [32]，以便根据文档自动部分推断模式效应和行为协议。

标准库：与此同时，我们正使用已实现的存根语言（第 IV-C 节）和自动存根生成（第 IV-D 节），集成 Pandas 的其他功能以支持数据准备、seaborn 的其他功能以支持可视化，以及 scikit-learn 的其他功能以支持传统机器学习。

可用性研究：待标准库实现到足够完善的程度后，我们将开展定量和定性比较，评估 Safe-DS 与 Python 解决现实问题的能力。

不同视图：目前，适用于 Safe-DS 的 Visual Studio Code 扩展仅提供文本视图。我们还计划使用 GLSP[10] 实现图形视图（第 V 节）。

IX. 结论

本文关注的是数据科学应用的需求，而非数据科学库。我们介绍了 Safe-DS（第 III 节），这是一种用于开发数据科学流水线的领域专用语言。它将 Python 数据科学库的表达能力与编译时错误检测带来的安全性结合起来。

Python 代码的类型安全集成基于存根的创建（第 IV-C 节），这些存根描述外部函数的接口，并在很大程度上自动从库代码和 文档中提取类型（第 IV-D 节）。Safe-DS 的当前实现已支持完整的传统静态类型系统：存根中使用显式类型，数据科学流水线中则使用类型推断。采用非标准类型概念（例如效应系统，用于检查表格数据集上操作的模式符合性，以及行为协议，用于检查调用顺序）也有助于静态地处理这些问题（第 IV-B 节）。

为了让编程技能有限的人也能开发数据科学流水线，并促进不同技能水平的人协作，我们的方法支持在该语言的文本和图形形态之间无缝转换（第 V 节）。总的来说，我们希望 Safe-DS 能帮助初学者踏上数据科学之旅。

致谢

本研究部分由德国联邦教育与研究部（BMBF）通过 Simple-ML 项目（资助号 01IS18054）资助。我们感谢所有曾为 Safe-DS GitHub 仓库作出贡献的人员，尤其是在模式检查和标准库方面的贡献。我们还感谢审稿人提出的大量建设性意见和建议。

[10]https://www.eclipse.org/glsp/

参考文献

[1] M. P. Robillard, 《是什么让 API 难以学习？来自开发者的答案》, 《IEEE Software》, 第 26 卷, 第 6 期, 第 27–34 页, 2009 年。[2] M. P. Robillard 和 R. DeLine, 《一项关于 API 学习障碍的实地研究》, 《Empirical Software Engineering》, 第 16 卷, 第 703–732 页, 2011 年。[3] M. Zibran、F. Eishita 和 C. Roy, 《有用但可用吗？影响 API 可用性的因素》, 载于《2011 年第 18 届逆向工程工作会议》, 2011 年 10 月, 第 151–155 页。[4] W. Wang 和 M. W. Godfrey, 《检测 API 使用障碍：一项关于 iOS 和 Android 开发者问题的研究》, 载于《2013 年第 10 届软件仓库挖掘工作会议》（MSR）, 2013 年, 第 61–64 页。[5] Kaggle。《2021 年 Kaggle 机器学习与数据科学调查》。 [在线] 2021 。可访问: 3 10

[6] https://kaggle.com/competitions/kaggle-survey-Python。Python。类型提示——对类型提示的支持——Python 文档第 7 版。 [在线]。可访问: https://docs.python.org/3/library/typing.html [7] I. Mierswa、M. Wurst、R. Klinkenberg、M. Scholz 和 T. Euler, 《YALE：复杂数据挖掘任务的快速原型开发》, 载于《第 12 届 ACM SIGKDD 国际知识发现与数据挖掘会议论文集——KDD '06》。美国宾夕法尼亚州费城：ACM 出版社, 2006 年, 第 935 页。[8] RapidMiner。RapidMiner Go。 [在线]。可访问: https://docs.rapidminer.com/latest/go/ [9] P. Molino、Y. Dudin 和 S. S. Miryala, 《Ludwig：基于类型的声明式深度学习工具箱》, 2019 年。 [在线]。可访问：https://arxiv.org/abs/1909 07930 。L. Reimann 和 G. Kniesel-Wunsche, 《通过软件工程技

[10] 术在应用机器学习中实现指导》, 载于《第 4 届艺术、科学与编程工程国际会议伴随论文集》, 丛书：<编程>'20。美国纽约州纽约：计算机协会, 2020 年 3 月, 第 7–12 页。[11] 维基百科贡献者。（2022 年）领域特定语言——维基百科，自由的百科全书。 https://en.wikipedia.org/w/index.php?title= Domain-specific language&oldid=1091164524。[12] J. D. Hunter, 《Matplotlib：二维图形环境》, 《Computing in Science & Engineering》, 第 9 卷, 第 3 期, 第 90–95 页, 2007 年。[13] M. L. Waskom, 《seaborn：统计数据可视化》, 《Journal of Open Source Software》, 第 6 卷, 第 60 期, 第 3021 页, 2021 年。 [在线]。可访问：https://doi.org/ 10 21105 03021 ./joss。Plotly

[14] Technologies Inc.。（）Plotly Python 绘图库。加拿大魁北克省蒙特利尔。 [在线]。可访问: 2015 https://plotly.com/python/ [15] pandas 开发团队。pandas-dev/pandas：Pandas。 [在线]。可访问：https://github.com/pandas-dev/pandas [16] F. Pedregosa、G. Varoquaux、A. Gramfort、V. Michel、B. Thirion、O. Grisel、M. Blondel、P. Prettenhofer、R. Weiss、V. Dubourg、J. Vanderplas、A. Passos、D. Cournapeau、M. Brucher、M. Perrot 和 E. Duchesnay, 《scikit-learn：Python 中的机器学习》, 《机器学习研究杂志》, 第 12 卷, 第 2825–2830 页, 2011 年。[17] L. Buitinck、G. Louppe、M. Blondel、F. Pedregosa、A. Mueller、O. Grisel、V. Niculae、P. Prettenhofer、A. Gramfort、J. Grobler、R. Layton、J. VanderPlas、A. Joly、B. Holt 和 G. Varoquaux, 《机器学习软件的 API 设计：来自 scikit-learn 项目的经验》, 载于《ECML PKDD 研讨会：数据挖掘与机器学习语言》, 2013 年, 第 108–122 页。[18] M. Abadi 等, 《TensorFlow：异构系统上的大规模机器学习》, 2015 年, 软件可从 tensorflow.org 获取。[在线]。可访问：https://www.tensorflow.org/ [19] F. Chollet 等, 《Keras》, https://keras.io, 2015 年。[20] A. Paszke、S. Gross、F. Massa、A. Lerer、J. Bradbury、G. Chanan、T. Killeen、Z. Lin、N. Gimelshein、L. Antiga、A. Desmaison、A. Kopf、E. Yang、Z. DeVito、M. Raison、A. Tejani、S. Chilamkurthy、B. Steiner、L. Fang、J. Bai 和 S. Chintala, 《PyTorch：命令式风格、高性能的深度学习库》, 载于《神经信息处理系统进展》第 32 卷, H. Wallach、H. Larochelle、A. Beygelzimer、F. d'Alch´e-Buc、E. Fox 和 R. Garnett 编。Curran Associates, Inc., 2019 年, 第 8024–8035 页。 [在线]。可访问:
h
t
t
p://
p
a
p
e
r
s.
n
eurips.cc/paper/9015-pytorch-an-i
mperative-style-high-performance-deep-learning-libra
ry.
PDF [21] T. Chen 和 C. Guestrin, "XGBoost：一种可扩展的树提升系统", 载于 第*22*届 *ACM SIGKDD* 国际知识发现与数据挖掘会议, KDD '16 系列。纽约
, 美国纽约州：ACM, 2016年, 第 785–794 页。 [在线]。可访问: http://doi.acm.org/ 10 1145 2939672 2939785 。/。I.

[22] Portugal、P. Alencar 和 D. Cowan, "大数据中机器学习领域特定语言初步调查", 载于 2016年 IEEE国际软件科学、技术与工程会议（SWSTE）, 2016年6月, 第 108–110 页。[23] C. Olston、B. Reed、U. Srivastava、R. Kumar 和 A. Tomkins, "Pig Latin：一种并非那么陌生的数据处理语言", 载于 2008 ACM SIGMOD国际数据管理会议论文集, SIGMOD '08 系列。纽约, 美国纽约州：计算机协会, 2008年6月, 第 1099–1110 页。[24] D. Breuker, "面向大数据分析的模型驱动工程——机器学习领域特定语言的探索性分析", 载于 2014年第47届夏威夷国际系统科学会议, 2014年1月, 第 758–767 页。[25] H. Khalajzadeh、M. Abdelrazek、J. Grundy、J. Hosking 和 Q. He, "BiDaML：支持终端用户数据分析的一套可视化语言", 载于 2019 IEEE国际大数据大会（BigDataCongress）, 2019年7月, 第 93–97 页。[26] H. Khalajzadeh、A. J. Simmons、T. Verma、M. Abdelrazek、J. Grundy、J. Hosking、Q. He、P. Ratnakanthan、A. Zia 和 M. Law, "BiDaML 实践：大数据分析应用需求的协作建模", 载于《软件工程新方法评估》, R. Ali、H. Kaindl 和 L. A. Maciaszek 编。Cham：Springer国际出版公司, 2021年, 第 106–129 页。[27] M. B. Kery、D. Ren、K. Wongsuphasawat、F. Hohman 和 K. Patel, "笔记本编程的未来是流动的", 载于《2020年 CHI计算系统中的人为因素会议扩展摘要》, CHI EA '20 系列。纽约, 美国纽约州：计算机协会, 2020年4月, 第 1–8 页。[28] T. Kluyver、B. Ragan-Kelley、F. P´erez、B. Granger、M. Bussonnier、J. Frederic、K. Kelley、J. Hamrick、J. Grout、S. Corlay、P. Ivanov、D. Avila、S. Abdalla、C. Willing 和 Jupyter 开发团队, "Jupyter 笔记本——可复现计算工作流的出版格式", 载于《学术出版中的定位与权力：参与者、代理人与议程》, F. Loizides 和 B. Scmidt 编。IOS出版社, 2016年, 第 87–90 页。 [在线]。可访问: https://eprints.soton.ac.uk/403913/ [29] X. He、K. Zhao 和 X. Chu, "自动化机器学习：最新进展综述", 《知识型系统》, 第 212 卷, 文章编号 106622, 2021年1月。[30] A. Jovic、K. Brkic 和 N. Bogunovic, "通用数据挖掘自由软件工具概览", 载于 2014年第37届国际信息与通信技术、电子与微电子大会（MIPRO）, 2014年, 第 1112–1117 页。[31] G. Nguyen、S. Dlugolinsky、M. Bob´ak、V. Tran、 ´A. L´opez Garc´ıa、I. Heredia、P. Mal´ık 和 L. Hluch´y, "面向大规模数据挖掘的机器学习与深度学习框架及库：综述", 《人工智能评论》, 第 52 卷, 第 1 期, 第 77–124 页, 2019年6月。[32] L. Reimann 和 G. Kniesel-W¨unsche, "通过半自动化 API 封装提高机器学习 API 的易学性", 载于 2022年 IEEE/ACM第44届软件工程国际会议：新思想与新兴成果（ICSE-NIER）, 2022年5月, 第 46–50 页。[33] J. M. Lucassen 和 D. K. Gifford, "多态效应系统", 载于第15届 ACM SIGPLAN-SIGACT编程语言原理研讨会论文集, POPL '88 系列。纽约, 美国纽约州：计算机协会, 1988年1月, 第 47–57 页。[34] D. Marino 和 T. Millstein, "一种泛型类型与效应系统", 载于第4届国际语言设计与实现类型研讨会论文集, TLDI '09 系列。纽约, 美国纽约州：计算机协会, 2009年1月, 第 39–50 页。[35] F. Plasil 和 S. Visnovsky, "软件组件的行为协议", 《IEEE软件工程汇刊》, 第 28 卷, 第 1056–1076 页, 2002年。[36] Y. Mao、D. Wang、M. Muller、K. R. Varshney、I. Baldini、C. Dugan 和 A. Mojsilovi´c, "数据科学家如何与领域专家开展科学协作：寻找正确答案, 还是提出正确问题？", 《ACM人机交互汇刊》, 第 3 卷, GROUP, 第 237:1–237:23 页, 2019年12月。[37] A. X. Zhang、M. Muller 和 D. Wang, "数据科学工作者如何协作？角色、工作流与工具", 《ACM人机交互汇刊》, 第 4 卷, CSCW1, 第 1–23 页, 2020年5月。[38] LSP贡献者。语言服务器协议官方页面。 [在线]。可访问: https://microsoft.github.io/language-server-protocol/