山东工商学院
Shandong Technology and Business University

大模型驱动的数据处理与数据分析

邓富文

计算机科学与技术学院

July 22, 2025

目录

引言：数据分析的困境

想象一下这个熟悉的场景：数学建模竞赛的倒计时已经开始，时钟在墙上无情地跳动。你和队友们正围坐在一台电脑前，屏幕上显示着刚刚下载的赛题数据——一个充满了缺失值、异常格式和潜在噪音的庞大 CSV 文件。

你们的目标是清晰的：在最短的时间内理解数据、发现规律、建立模型。然而，现实的路径却布满了障碍。

- “注册时间”这一列是文本格式，怎么转换成日期？
- 这个特征的缺失值，应该用均值还是中位数填充？
- 快，画一个散点图看看这两个变量有没有关系！

引言：传统分析的痛点与 LLM 的出现

传统数据分析的困境

每一个简单的想法，都意味着一行行代码的编写、调试和运行。思维在“分析问题的逻辑”和“实现功能的语法”之间不断切换，消耗着宝贵的竞赛时间与认知资源。我们需要投入大量精力去告诉计算机“如何做”，而非专注于“做什么、为什么做”。

大语言模型（LLM）的颠覆

大语言模型（Large Language Models, LLM）作为人工智能领域的璀璨明珠，其强大的自然语言理解和代码生成能力，正以前所未有的方式颠覆着数据科学领域。它如同一位精通多种编程语言的“超级翻译官”，能够精准地理解我们的自然语言指令，并将其瞬间转化为计算机可以执行的、结构化的代码。

引言：LLM 在数据分析中的优势

1. **效率提升**：当你提供了数据分析需求时，LLM 会立即为你生成并执行相应的代码。这种“所思即所得”的交互模式，将数据分析的效率提升了数个量级，让你能更快地进行数据探索。

2. **门槛降低**：你不再需要成为一个精通 MATLAB 或 Python 所有函数和参数的编程专家，团队中对建模有深刻理解但编程稍弱的成员，同样可以通过自然语言直接与数据对话，让团队的集体智慧得以最大化发挥。

3. **创新启发**：当数据探索的成本变得极低时，你可以自由地提出各种探索性的问题，例如：“数据中是否存在有趣的聚类?”“哪些变量的组合对目标影响最大?”。LLM 能够快速生成多样化的可视化图表和分析结果，帮助你迅速验证假设、捕捉灵感。

LLM 驱动的数据分析工具：Pandas AI

Pandas AI 简介

在众多 LLM 驱动的数据分析工具中，`Pandas AI` 无疑是最具代表性的先行者之一。它的核心使命简单而强大：为最受欢迎的数据分析库 `pandas` 安装一个“智慧大脑”，使其能够理解自然语言，将你从繁琐的代码记忆中解放出来。

核心体验

你不再需要去查阅 `pandas` 的官方文档来确定一个函数的具体参数，而是像与一位数据分析专家同事交谈一样，直接告诉它你的需求。

Pandas AI: 工作原理浅析

`Pandas AI` 的工作流程如同一座桥梁，连接了人类的自然语言与计算机的结构化代码：

1. **指令输入 (Prompt)：** 通过 `.chat()` 方法，用一句清晰的指令向 `Pandas AI` 提出你的分析请求。
2. **LLM 转译 (LLM Translation)：** `Pandas AI` 将指令连同 DataFrame 的元数据（列名、数据类型等）一同发送给其配置的大语言模型。
3. **代码生成 (Code Generation)：** LLM 凭借其强大的代码生成能力，将自然语言指令“翻译”成一段可执行的、专门针对该 DataFrame 的 `pandas` Python 代码。
4. **安全执行与审查 (Safe Execution & Validation)：** 在执行代码前，`Pandas AI` 会进行一系列安全检查，防止潜在的恶意代码执行。然后，它运行生成的代码。
5. **结果输出 (Result Output)：** 最后，将代码的执行结果返回给用户。

Pandas AI: 实践

环境搭建

- 安装 `pandasai`、`pandasai-litellm` 库（建议在 Python 3.10 或 3.11 版本安装）。
- 准备大模型：`Pandas AI` 支持多种 LLM，可以根据自己的情况选择。
- 创建大模型 API 密钥，以便 `Pandas AI` 能够调用它。

采用自然语言进行数据分析

具体的 Python 代码实现，请参考随附文件：

代码文件：`pandasai.ipynb`

LLM 驱动的数据分析工具：Lida

Lida 简介

LIDA 是微软（Microsoft）开源的一个用于生成数据可视化和信息图的库。LIDA 可以与任何编程语言和可视化库一起使用，例如 matplotlib、seaborn、altair、d3 等，支持多个大型语言模型提供商（包括 OpenAI、Azure OpenAI、PaLM、Cohere、Huggingface，其他大模型暂不支持）。

Lida: 核心功能

LIDA 的核心价值体现在以下几个方面：

- **数据摘要 (Summarization):** `Lida` 能自动读取数据集，并生成一份关于数据语义、结构和潜在分析点的“体检报告”。
- **目标探索 (Goal Exploration):** 这是 `Lida` 的独特之处。它可以基于数据摘要，自动推荐一系列有意义的分析目标（Goals），比如“探究 XX 和 YY 之间的关系”，启发你的分析思路。
- **可视化生成 (Visualization Generation):** 你可以从推荐的目标中选择，或者直接用自然语言描述你想要的图表，`Lida` 会为你生成代码，并支持多种后端。
- **信息图呈现 (Infographics):** 能将分析结果以更高级、更具视觉冲击力的信息图（Infographic）形式展示，适合用于最终的论文插图。

Lida: 实践

环境搭建

首先确保 Python 版本是 3.10 或更高版本。之后可通过 pip 包管理器安装 `lida`。同时，LIDA 依赖 `llmx` 和 `openai` 包，需要单独安装它们。

使用方法

使用 LIDA 能够实现自动化的探索性数据分析。具体代码实现请参考官方文档，此处仅示意其调用过程。

构建智能数据分析工作流：从“一问一答”到“问题链”

我们已经体验了 AI 数据分析工具的强大威力。通过一句简单的自然语言指令，就能为我们完成复杂的数据筛选、分析、可视化等工作。

然而，在真实的数据分析任务中，我们面对的往往不是孤立的任务点，而是一个需要多步骤、多工具协作才能完成的复杂**“问题链”**。

为了打破这种局限，我们需要从“使用工具”上升到“创造工人”的层面。这个“工人”，就是我们要介绍的核心概念——**智能体（Agent）**。

工作流的基本理念：智能体 (Agent) I

你可以将一个 Agent 想象成一个虚拟的、高度自主的“智能数据分析师”。它被赋予了一个总体的、宏观的目标，然后它会自己想办法去完成。

一个合格的 Agent，必须具备三个核心要素：

- **思考能力 (Reasoning):** 这是 Agent 的“大脑”，通常由一个强大的大语言模型（LLM）担当。它负责理解你的最终目标，并将其分解成一系列合乎逻辑、可执行的子任务。
- **工具箱 (Tools):** 这是 Agent 的“双手”，是其可以使用的各种能力的集合。每个工具都是一个专门的函数，能完成一项特定任务（如数据分析、网络搜索、文本总结）。

工作流的基本理念：智能体 (Agent) II

- **行动计划 (Planning):** 这是 Agent 的“决策循环”。它根据“思考”结果，从“工具箱”中选择最合适的工具来执行子任务。这个“思考 → 行动 → 观察”的循环会不断重复，直到最终目标达成。

LangChain 简介：构建 Agent 的开源框架

什么是 LangChain?

`LangChain` 是一个功能强大且极受欢迎的开源框架。它的核心使命就是将开发 LLM 应用过程中的各种复杂组件（如模型调用、数据连接、任务链接、记忆管理等）抽象化、标准化，让你能够像拼搭乐高一样，快速、灵活地将这些“积木”拼接起来，创造出功能强大的应用程序。

LangChain 对数据分析的意义

`LangChain` 让我们能轻松地从“工具使用者”转变为“工作流设计师”。它提供了构建智能数据分析 Agent 所需的一切基础构件。

LangChain 核心组件

LLMs

大模型是 Agent 的 “大脑”，为 Agent 提供了思考能力。`LangChain` 提供了一个统一的 `LLM` 类，它抹平了不同大模型提供商之间的 API 差异。无论你背后使用的是 GPT-4、Gemini 还是 Deepseek，在 `LangChain` 的代码层面，你都可以用完全相同的方式来调用它们。

Chains (链)

“Chain” 是 `LangChain` 中最基础、最核心的执行单元。它的作用就是像链条一样，将多个组件按顺序串联起来，形成一个逻辑清晰的调用流程。它负责接收输入，将其传递给链中的第一个组件，然后将该组件的输出作为下一个组件的输入，依次进行。最基础的链是 `LLMChain`。

LangChain 核心组件

Tools (工具)

“工具” 是 Agent 可以使用的具体能力的封装。每一个工具都是一个函数，它被赋予了清晰的名称和功能描述，以便 Agent 能够理解并决定何时使用它。例如：`python_repl`（执行 Python 代码）、`google-search`（网络搜索），甚至可以自定义工具。

LangChain 核心组件

Agents (智能体)

Agent 是驱动这一切的决策核心。它本身也由一个 LLM 驱动，但它的任务不是直接回答问题，而是选择工具。其工作遵循一个名为 “ReAct”（Reason + Act，思考 + 行动）的逻辑循环：

1. **Reason (思考):** 观察目标和现状，思考下一步做什么。
2. **Act (行动):** 选择最合适的工具并提供输入。
3. **Observe (观察):** 执行工具并观察返回结果。
4. **Repeat (重复):** 带上新结果回到第一步，直到目标完成。

搭建 Python 数据分析智能体

环境配置

首先，确保你已经安装了必要的库: `langchain`, `langchain-community`, `langchain-experimental`, `langchain-openai`, `openai`, `pandas`, `tabulate`。

编写 Python 代码初始化 Agent

我们使用 `LangChain` 的一个开箱即用的功能：`Pandas DataFrame Agent`。这个 Agent 专门为与 `pandas` 数据交互而设计，它不仅能理解你的问题，更能自主地编写并执行 Python 代码来回答这些问题。

代码文件：`agent.ipynb`

执行多步推理任务

一个更复杂的任务

请计算各洲的人均 GDP，找出人均 GDP 最高的洲，然后画出该洲所有国家的 GDP 条形图。

这个任务需要多个步骤才能完成，单一的查询无法解决。Agent 通过其“思考链”(Chain of Thought)，自主地完成任务分解和代码执行。

查看 Agent 的思考过程

由于在创建 Agent 时设置了 `verbose=True`，运行 `.invoke()` 时，你会在控制台看到 Agent 完整的思维链。

代码文件：`agent.ipynb`

设计自定义的分析工作流

为什么要自定义?

我们已经掌握了如何使用预置的 `Pandas Agent`，但要应对数据分析中的各种挑战（例如执行特定的统计检验、保存图表到文件），我们需要掌握自定义数据分析工作流的方法。

创建自定义工具的核心

在 `LangChain` 中，“工具”的本质是一个带有清晰描述的 Python 函数。Agent 的“大脑”（LLM）并不理解函数内部的逻辑，但它能读懂函数的**描述文档 (docstring)**。这个描述是 Agent 决定“何时”以及“如何”使用该工具的唯一依据。

自定义工具案例

案例 1：一个进行 T 检验的统计工具

在分析数据时，我们经常需要比较两组数据的均值是否存在显著差异。T 检验就是实现这一目标的标准统计方法。我们可以创建一个工具来实现它。

- `@tool` **装饰器**: LangChain 中工具的标志。
- **清晰的文档字符串**: 整个工具的灵魂，告诉 Agent 工具的功能、何时使用、输入格式。

代码文件： `custom_tool.ipynb`

案例 2：一个保存图表的工具

LangChain 自带的 `Pandas Agent` 虽然能自动生成图表，但是不能帮助我们保存图表到文件。因此，我们可以自己设计一个保存图表工具。

代码文件： `custom_tool.ipynb`

创建自定义 Agent 并执行复杂任务 I

创建自定义 Agent

现在我们拥有了：

1. 一个预置的、强大的 `Pandas Agent`（作为基础工具）。
2. 两个我们自定义的专业工具：`t_test_tool` 和 `save_plot_tool`。

我们将这些工具组合起来，创建一个更强大的“总管 Agent”，让它来调度所有工具完成复杂任务。

创建自定义 Agent 并执行复杂任务 II

执行复杂指令

请分析我提供的数据，完成以下任务：*1.* 检验产品 `A_` 评分和产品 `B_` 评分这两组数据的均值是否存在显著差异。*2.* 在数轴上绘制点展示产品的评分... 不要调用 *plt.show()*。*3.* 将生成的图表保存为 `'ratings_analysis.png'`。*4.* 最后，总结你所有的发现。

完整代码请见： `custom_tool.ipynb`

如何提出一个好问题

在我们与大模型的每一次交互中，我们输入的指令/提示词（Prompt）是决定输出质量的唯一变量。

原则一：清晰性与明确性

你必须用最直接、最没有歧义的语言告诉它具体要做什么。

- **反面案例（模糊）：** “分析一下这份数据。” / “哪个产品最好？”
- **正面案例（明确）：** “计算每个产品类别的总销售额，并按总销售额从高到低排序。” / “找出过去一个季度中，销售额环比增长率最高的前 5 个产品。”

如何提出一个好问题

原则二：提供上下文

为你的问题提供充足的上下文，能帮助 AI 更好地理解你的目标。

- **无上下文：** “计算增长率。”
- **有上下文：** “我正在进行年度销售数据的分析... 请计算每个销售区域 2023 年相对于 2022 年的销售额同比增长率。”

如何提出一个好问题

原则三：迭代优化与分解任务

不要指望用一个“完美”的 Prompt 一蹴而就。更高效的方式是进行多轮、渐进式的沟通。将一个复杂的分析任务分解成一系列更小、更简单的步骤，然后逐步引导 AI 完成。

- **技巧：**把 AI 当作你的副驾驶，你来把握方向盘（提出总体目标和关键步骤），让它来处理具体的驾驶操作（执行代码）。

原则四：角色扮演

这是一个非常有效的高级技巧。你可以通过指令，要求 AI 扮演一个特定的角色。

- **无角色指令：**“总结一下你的发现。”
- **角色扮演指令：**“你现在是一位销售数据分析专家。请用简洁、专业的语言，总结你从数据中发现的三个最关键的洞察。”

AI 工具是一把“双刃剑”

AI 工具无疑是我们在数据分析战场上的一把锋利无比的利剑。然而，它也是一柄“双刃剑”。剑刃的锋利面，是其无与伦比的效率；而剑柄的另一面，则刻着使用者必须承担的责任与潜在的风险。

知其然，更要知其所以然

AI 最诱人的地方在于，它能让你在不完全理解底层原理的情况下得到答案。但这是一个极其危险的陷阱。

- **将 AI 视为加速器，而非替代品：**分析的思路、模型的选择、变量的取舍，必须源于你和团队的智慧。
- **审查每一行生成的代码：**当大模型返回结果时，务必查看它背后实际执行的代码。如果你看不懂，那就等于将自己模型的命运交到了一个“黑箱”手中。

代码审查与结果验证

代码审查是不可或缺的环节

AI 生成的代码不总能 100% 正确，也不总能 100% 高效。必须做到：

- **检查逻辑的准确性：** AI 是否正确理解了分析逻辑？（例如，剔除异常值的方法是否合理？增长率是同比还是环比？）
- **评估方法的合理性：** AI 选择的函数或方法是否是最佳实践？（例如，代码是否做了向量化，还是使用了低效的循环？）
- **验证结果的正确性：** 对于关键的计算结果，务必进行手动抽样验算。

信任但核实 (Trust but Verify)

红线警告：学术诚信与代码提交注意事项 I

这是最重要的一点，请所有参赛者严格遵守！

- **严禁使用 AI 直接生成论文内容：**你的竞赛论文必须完全由团队成员亲自撰写。使用 AI 生成任何段落、摘要、结论等文字内容，都属于严重的学术不端行为。
- **严禁在论文中提及 AI 工具的使用：**在你的论文正文、附录或任何提交材料中，都不应出现“我们使用了 Pandas AI”、“通过 LangChain Agent 分析”等字样。
- **严禁提交调用 AI 工具的源代码：**你最终提交的代码附件，必须是纯粹的、可独立运行的代码。代码中绝不能包含任何对 `pandas-ai`、`langchain` 等库的调用，例如 `.chat()` 或 `.invoke()` 等方法。

红线警告：学术诚信与代码提交注意事项 II

正确做法

AI 是你过程中的 **“脚手架”**，而不是你最终的 **“交付物”**。竞赛考察的是你作为建模者的综合能力，在最终提交成果时，你必须拆除所有“脚手架”，向评委展示一个完全由你亲手搭建的、坚实可靠的“建筑”。

练习 4（选做）

本练习题不做强制要求，对大模型应用感兴趣的同学可以选做。

要求

使用 `PandasAI` 或 `LangChain` 作为过程工具，分析 `book_sales.csv`。你需要记录下你与 AI 交互的关键问题（Prompt），并尝试整理出不依赖 AI 库的纯净 Python 代码。

1. **（使用 PandasAI）：**在所有书籍中，哪一本书的销售额最高？绘制图表并给出结论。
2. **（使用 LangChain Agent）：**这是一个多步任务：首先，找出哪一天的总销售额最高；然后，列出在那一天销售的所有书籍名称。

感谢观看！