基于OpenDiger的开源项目健康度评估与可视化

Created @2026年1月13日 20:16

闫涵 10241900402

一、项目背景与研究动机

随着开源软件在软件工程中的广泛应用，我们可以看到各大开源平台上都有数量巨大的开源项目，而哪些是可用度比较高的呢？于是评估一个开源项目是否“健康”成为一个具有现实意义的问题。本项目是课程《数据科学与工程导论》的期末作品，目的在于

> 综合运用 Python 编程相关知识，实现一个从数据读取、分析建模到可视化展示的完整系统。

所以，本项目基于 OpenDigger 提供的开源项目指标数据，、尝试构建一个**开源项目健康度分析系统**，对项目的运行状态进行定量度量与分析。

二、项目整体思路

1、 整体流程设计

本项目整体实现流程如下：

1. **数据获取**：读取 OpenDigger 提供的项目 JSON 指标数
2. **数据处理**：进行数据清洗，并将不同维度的指标转换为**时间序列**
3. **健康度建模**：设计多维指标（活跃度、影响力、协作度、维护压力）、指标归一化、计算综合健康值
4. **结果分析**：从**单项目纵向分析（时间趋势、健康结构雷达图）**和**多项目横向对比（健康值排序、健康值分布、指标关系散点图）**两个角度分析项目状态
5. **可视化与交互**：通过 Web 界面直观展示分析结果

在前端展示选择 Streamlit Web应用，原因在于其能够**使用纯 Python 构建交互式网页应用**，非常好用

三、最初版本的实现

1、 初始目标

在最初版本中，项目的目标非常明确：

> 针对单个开源项目，计算一个简单的健康度指标，并进行基本可视化分析。

2、 数据读取与解析

OpenDigger 数据以 JSON 文件形式存储，因此首先需要实现 JSON 文件的读取与解析。

```
import json

withopen("activity.json","r", encoding="utf-8")as f:
    data = json.load(f)
```

由于 JSON 中的数据结构通常为：

```
{
"2023-01":{"count":120},
"2023-02":{"count":98}
}
```

因此需要进一步提取数值并构造时间序列：

```
values = []
for vin data.values():
    values.append(sum(v.values()))
```

3、最初的健康度定义

在最初版本中，健康度主要由两个指标构成：

- 活跃度：项目开发是否活跃

- 影响力：外部关注程度

健康度的计算方式为：

```
health =0.5 * activity +0.5 * openrank
```

4、 初始可视化（Matplotlib）

```
import matplotlib.pyplotas plt

plt.plot(activity_series)
plt.title("Project Activity Trend")
plt.show()
```

通过折线图观察项目活跃度随时间的变化。

但是，随着项目的推进，最初版本暴露出、指标维度过少、只能分析单个项目，缺乏横向比较、交互性较差等缺点。因此，项目进入持续优化阶段。

四、数据处理优化

为了将原始的 OpenDigger JSON 指标数据转化为可用于分析的数值，不仅要是对数据进行清洗、处理，更要把文件中按时间存储的数据抽取为列表，然后进行分析整合。所以进行了以下三个步骤：

1、时间序列抽取：将原始数据变为数值序列

关键在于要处理JSON不同的格式结构，并且屏蔽非数值字段

```
def parse_series(path):
    if not path or not os.path.exists(path):
        return None

    with open(path, "r", encoding="utf-8") as f:
        data = json.load(f)

    series = []
```

```
    for v in data.values():
        if isinstance(v, dict):
            series.append(sum(x for x in v.values() if isinstance(x, (int, float))))
        elif isinstance(v, (int, float)):
            series.append(v)

    return series if series else None
```

2、数据处理（缺失与异常）

要将缺失指标进行识别，防止程序中断

```
def safe_mean(series, window):

    if not series or len(series) == 0:
        return 0
    return np.mean(series[-window:])
```

3、降噪处理

为了防止数据过于冗杂，我们只关注项目近期波动和异常值影响

```
window = 12
# 最近 12 个月
activity_score = safe_mean(activity_series, window)
impact_score = safe_mean(impact_series, window)
```

五、健康度模型的改进

1、多维健康度设计思想

考虑到日常对开源项目引用时会参考的数据，以及其他类似分析会有的指标，本项目将健康度拆分为四个维度：

健康维度	含义	原始数据文件（JSON）	数据说明
活跃度	是否持续开发	`activity.json` 或 `commit*.json`	提交、开发行为强度
影响力	是否被社区关注	`openrank.json` 或 `stars.json`	社区关注与传播

健康维度	含义	原始数据文件（JSON）	数据说明
协作度	是否有多人参与	`contributors.json` / `pull*.json`	协作与参与度
维护压力	维护是否健康	`issues*.json`	Issue 积压情况（负向）

2、 指标归一化与对数变换

不同指标数量级差异较大，如果直接使用，会出现图表“长尾”的现象，并不便于观察。因此使用 `log1p` 进行压缩：

```
import math

A_n = math.log1p(A)
I_n = math.log1p(I)
```

还要注意到负数问题，Issue 数为负向指标，在模型中进行反向处理：

```
、maintenance_score =1 - M_n / (M_n +1)
```

3、 最终健康度模型

```
health = (
    w_a * A_n +
    w_i * I_n +
    w_c * C_n +
    w_m * maintenance_score
)
```

六、交互式前端的实现

因为不想局限于只用脚本分析，想让这个模型供更多人使用，在尝试过多个平台后，选择引入 Streamlit，将分析过程封装为一个 Web 应用，使结果更直观。

1、 项目自动扫描

在刚开始的尝试中，由于不同文件json所在目录等级不一，无定法统一调度，所以使用自动扫描。避免了对目录层级的硬编码，提高了系统可操作性。

```
projects = []

for root, _, files in os.walk(DATA_DIR):
    # 若当前目录下存在 JSON 指标文件，则视为一个可分析项目
    if any(f.endswith(".json") for f in files):
        projects.append(root)

# 去重并排序，保证结果稳定
projects = sorted(set(projects))
```

2、单项目深度分析模块

该模块包括：健康值 KPI 显示、多指标时间趋势图、健康结构雷达图、Issue 维护压力分析

```
# 计算健康值
health = (
    w_a * A_n +
    w_i * I_n +
    w_c * C_n +
    w_m * maintenance_score
)

# 核心指标
c1, c2, c3, c4 = st.columns(4)
c1.metric("健康值", f"{health:.2f}")
c2.metric("活跃度", f"{A:.1f}")
c3.metric("影响力", f"{I:.1f}")
c4.metric("协作度", f"{C:.1f}")

#多指标时间趋势分析
df_trend = pd.DataFrame({
    "Activity": activity[-window:],
    "Impact": impact[-window:],
    "Collaboration": collaboration[-window:]
})
```

```
st.line_chart(df_trend)

#健康结构雷达图
labels = ["Activity", "Impact", "Collaboration", "Maintenance"]
values = [A_n, I_n, C_n, maintenance_score]
values += values[:1]

angles = np.linspace(0, 2 * np.pi, len(labels) + 1)

fig = plt.figure()
ax = plt.subplot(111, polar=True)
ax.plot(angles, values)
ax.fill(angles, values, alpha=0.3)
ax.set_thetagrids(angles[:-1] * 180 / np.pi, labels)

st.pyplot(fig)

#Issue 维护压力趋势分析
fig = plt.figure()
plt.plot(issues[-window:])
plt.xlabel("Time")
plt.ylabel("Issues")
plt.grid(True)

st.pyplot(fig)
```

3、横向对比分析：从单点到多点

为了从整体角度观察排序，引入横向对比模块：

- 计算所有项目健康值
- 排序并展示 Top-N 项目
- 分析健康度整体分布

```
#Top-N 项目健康度排序展示
top_n = 20

top_projects = (
```

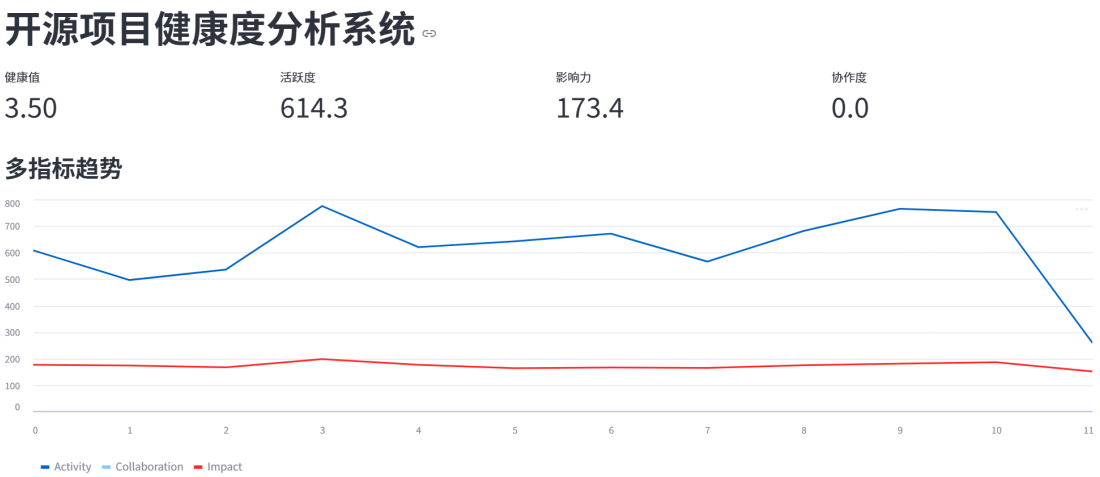
```
    df.sort_values("Health", ascending=False)
      .head(top_n)
)
st.bar_chart(
    top_projects.set_index("Project")["Health"]
)

#健康度整体分布分析
fig = plt.figure()
plt.hist(df["Health"], bins=20)
plt.xlabel("Health Score")
plt.ylabel("Number of Projects")
plt.grid(True)

st.pyplot(fig)
```

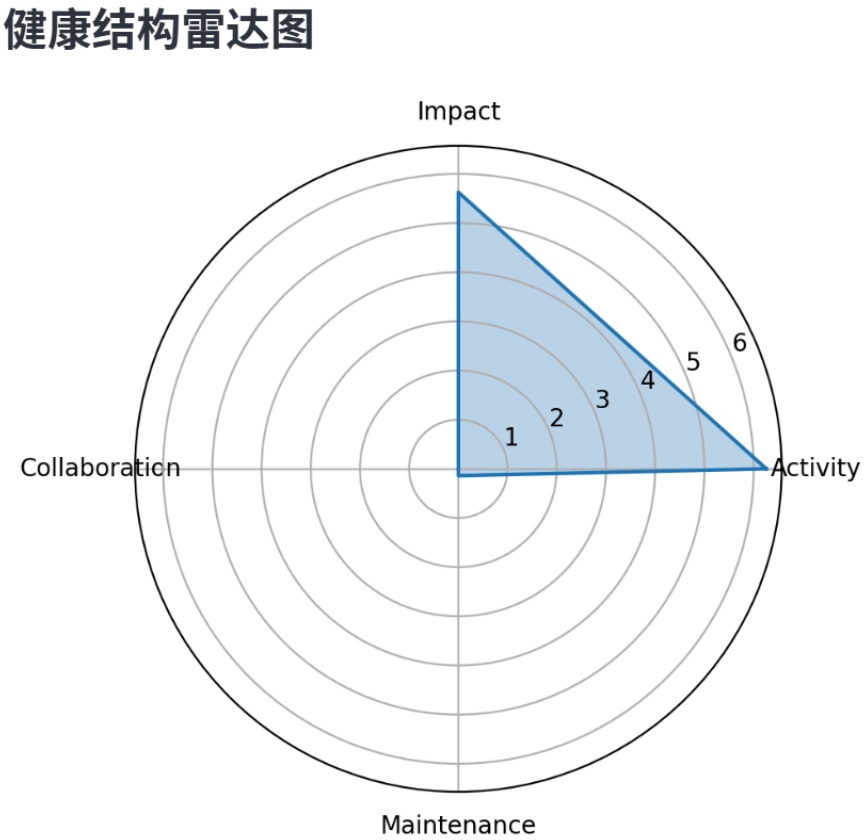
可以横向比较不同项目的健康度对比情况、整体分析

七，网页展示（部分）

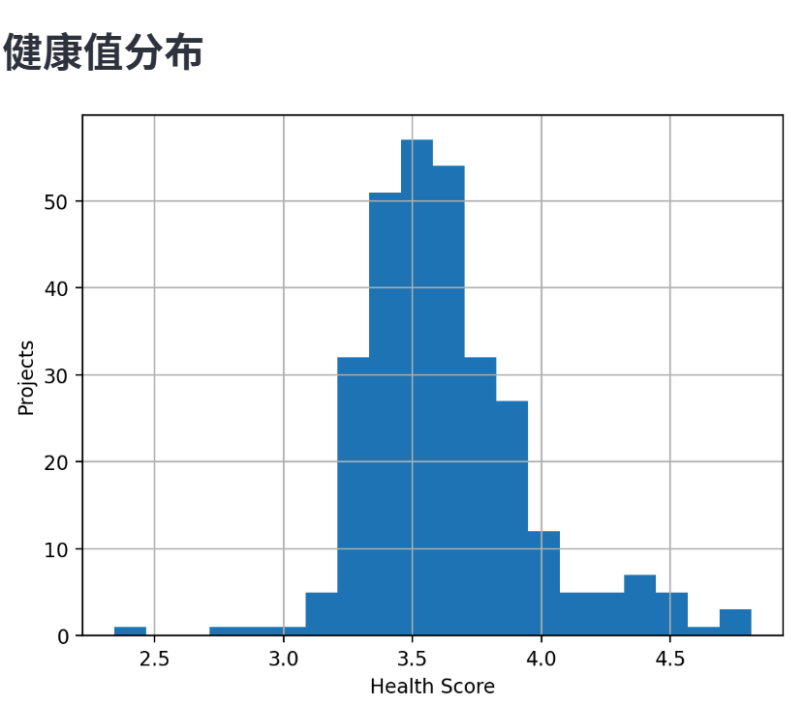
边栏

单项目纵向分析





多项目横向分析



八、总结与反思

本项目基于 Python 对OpenDigger开源项目健康度进行了系统分析，从最初的单指标脚本实现，逐步发展为一个支持多维健康度建模和横向对比分析的交互式分析系统。

项目通过引入活跃度、影响力、协作度和维护压力等多个维度，刻画了开源项目的运行状态，并在工程实现上实现了整个系统。

但在实现过程中，还有以下不足：

1. 当前健康度模型采用人工设定权重，仍具有一定主观性，具体怎样的权重比例更准确，还需要进一步通过大量数据进行探索。
2. 指标维度主要集中在行为和社区层面，尚未覆盖代码质量等因素。有一些代码可能质量很高，足额在社区中不够突出，未来还需要参考代码质量或者编程思维这部分。
3. 分析结果以可视化为主，缺乏自动化解释，这一点可以考虑后期接入MaxKB此类AI，帮助用户分析。

一个完整的数据分析系统，往往需要经历不断迭代和结构优化的过程。未来可以引入更丰富的指标体系和更智能的分析方法，进一步提升健康度评估的准确性和系统实用价值