实验作业1

本实验旨在让同学初步了解软件工程的研究流程，包括数据获取、特征工程、传统机器学习方法等。实验任务聚焦于 GitHub 项目中的 Pull Request（PR），分两类目标：预测PR处理时间长短与预测结局。

一、实验目标与背景

- **任务一：预测 Pull Request 处理时间长短**
  - **定义（任选其一）**：
    i. **首个反馈时长 (Time-to-First Response, TFR)**：从 PR 创建到第一次评论或评审意见的间隔。
    ii. **关闭时长 (Time-to-Close, TTC)**：从创建到 PR 被合并或关闭的间隔。
  - **意义**：合理预估处理时长可以帮助维护者优先处理紧急 PR，提高代码评审效率。
- **任务二：预测 Pull Request 合入与否**
  - **定义**：对每个 PR 进行二分类，判断其是被合并（merged）还是在审查流程后关闭但未合并（closed）。
  - **意义**：帮助维护者提前识别可能被拒绝的PR，从而改进贡献者的提交或分配评审资源。

二、数据集与特征

2.1 原始数据集

课程提供单一项目的 PR 数据，包括 PR 基础信息（创建时间、关闭时间、标题、正文）以及部分代码统计信息。为保证实验的公平性和防止数据泄漏，按时间顺序分割训练集和测试集。例如将训练集设为**项目开始**至 2021 年 5 月 31 日、测试集设为 2021 年 6 月 1 日至 2022 年 6 月 15 日。在进行实验时，应使用类似的时间切分策略，而不是随机划分样本。

2.2 特征工程

序号	特征名称	特征描述
1	assignees	The number of assignees (ie @xxx).
2	has_test	Whether the function name contains the word test?
3	has_bug	Whether the description contains a bug?
4	has_feature	Whether the description contains a feature?
5	has_improve	Whether the description contains a improve?
6	has_document	Whether the description contains a document?
7	has_refactor	Whether the description contains a refactor?
8	directories	The number of directories modified.
9	language_types	The number of programming language types used.
10	file_types	The number of file types in the RR.
11	lines_added	The number of lines of code added.
12	lines_deleted	The number of lines of code deleted.
13	segs_added	The number of segments of code added.
14	segs_deleted	The number of segments of code deleted.
15	segs_changed	The number of segments of code changed.
16	files_added	The number of files added.
17	files_deleted	The number of files deleted.

18	files_changed	The number of files changed.
19	file_developer	The number of developers who changed files.
20	change_num	The average number of commits of a RR submitted by the author.
21	files_modified	The number of times files were modified before.
22	is_core_member	Whether the author is a core member of the project?
23	commits	The number of commits.
24	prev_PRs	The number of PRs created by the author.
25	title_words	The number of words in the title.
26	body_words	The number of words in the body.
.....	.....	.....

- **二元文本标记**：如 has_test、has_bug、has_feature、has_improve、has_document、has_refactor 等，分别表示标题或正文中是否包含相应关键词。这些特征可以用简单的字符串查找实现。

```python
key_words = ['bug', 'document', 'feature', 'improve', 'refactor']
has_key_words = []
for item in process_data:
    tmp = []
    if item[body_index] is None:
        tmp = [0, 0, 0, 0, 0]
    else:
        for k in key_words:
            if k in item[body_index]:
                tmp.append(1)
            else:
                tmp.append(0)
    has_key_words.append(tmp)
```

- **目录与文件结构**：包括修改的目录数 (directories)、所涉及的编程语言类型数 (language_types) 与文件类型数 (file_types)。同学可以根据 PR 中修改文件的路径、扩展名统计这些值。

```python
def get_file_directory_num(pr_file_dict):
    """
        pr_file_dict:
        {
         733:  {
             file_name:  {
             'guacamole/src/main/java/org/apache/guacamole/rest/usergroup/UserGroupObjectTranslator.java': {'@@ -57,7 +57,7 @@ public void filterExternalObject(UserContext userContext, APIUserGroup object)\n ......'
             }
         }
        返回的dict为{id: 涉及到的文件路径数量}
        """
    re_dict = {}
    for key in pr_file_dict.keys():
        pr_number = key
        directory_num = 0
        for file_name_key in pr_file_dict[pr_number].keys():
            file_first_split = str(file_name_key).split('/')
            directory_num = directory_num + file_first_split.__len__() - 1
        re_dict[key] = directory_num
    return re_dict
```

- **修改规模特征**：如增加行数 (lines_added)、删除行数 (lines_deleted)、添加文件数 (files_added)、删除文件数 (files_deleted)、修改文件数 (files_changed) 等。

```
  "comments": 0,
  "review_comments": 0,
  "maintainer_can_modify": false,
  "commits": 1,
  "additions": 4,
  "deletions": 1,
  "changed_files": 1
}
```

- **作者经验与活跃度**：包括作者是否为项目核心成员 (is_core_member)、作者过往创建的 PR 数量 (prev_PRs)。

  课程只提供部分特征的计算示例代码，其余特征需要同学自行实现。

三、建模方法与评估指标

3.1 任务一：预测 Pull Request 处理时间长短

- **建模方法**：回归模型
- **评价指标**：平均绝对误差（MAE）、均方误差（MSE）、均方根误差（RMSE）、决定系数（ R^2 ）等

3.2 任务二：预测 Pull Request 合入与否

- **建模方法**：分类模型
- **评价指标**：准确率(Accuracy)、精确度(Precision)、召回率(Recall)、F1分数（Macro平均）等

四、难度梯度

为了确保每位同学都能从实验中受益，我们将任务分为三个难度层级：

1. **基础层（Level 1）**：使用给定的单项目数据及部分特征完成非神经网络基线模型，并报告实验过程与结果。（70 分）
2. **进阶层（Level 2）**：基于Level 1的工作，自行通过GitHub REST API获取至少两个项目的数据，进行数据清洗和特征提取后，重复Level 1的实验过程，并记录整个实验流程及其成果。（20 分）
3. **拓展层（Level 3）**：基于Level 2的工作，探索多种模型的效果，并进行特征消融实验。（10 分）

达到更高级别可以获得额外分数，但完成**基础层（Level 1）**是**实验合格的基本条件。**

五、数据获取流程提示

如果希望从多个项目获取 PR 相关数据。以下流程提供一个抓取思路：

1. **确定目标项目**：选择若干流行或活跃的开源仓库，要求 PR 数量大于 1000，如 yiisoft/yii2、tensorflow/tensorflow 或 pytorch/pytorch。
2. **获取 PR 列表**：使用 /repos/{owner}/{repo}/pulls?state=all 获取所有 PR。由于 API 返回的是分页数据，需要循环或递归请求。每个 PR 返回了创建时间、关闭时间、合入状态等基础字段。
3. **获取 PR 详细信息**：对每个 PR 调用 /repos/{owner}/{repo}/pulls/{pr_number} 以获取完整的字段，如标题、正文、提交数量、添加删除行数等。
4. **获取修改文件列表**：调用 /repos/{owner}/{repo}/pulls/{pr_number}/files 获取改动的文件、添加/删除行数等，用于统计目录数、文件类型数以及改动规模。
5. **获取评论和审核信息（可选）**：对于时间预测任务，需要获取首个评论或评审的时间，可以调用 /repos/{owner}/{repo}/pulls/{pr_number}/reviews。
6. **提取作者信息**：通过 /repos/{owner}/{repo}/contributors 获取作者是否为协作者、核心成员等信息。也可以统计作者历史 PR 数量作为经验特征。

7. **清洗与合并数据**：将上述信息合并成一条条 PR 记录，计算特征值并保存为 CSV 或数据库。
8. **时间切分**：按创建时间将数据分为训练集和测试集，注意保持测试集的时间晚于训练集。

以下是一个简化的代码示例，用来分页获取某仓库的所有 PR：

```python
import requests

GITHUB_TOKEN = "<your_token_here>"
owner = "tensorflow"
repo = "tensorflow"

headers = {
    "Authorization": f"token {GITHUB_TOKEN}",
    "Accept": "application/vnd.github+json"
}

prs = []
page = 1
per_page = 100  # GitHub API 最大支持 100 条/页

while True:
    url = f"https://api.github.com/repos/{owner}/{repo}/pulls"
    params = {"state": "all", "per_page": per_page, "page": page}
    response = requests.get(url, headers=headers, params=params)
    response.raise_for_status()
    data = response.json()
    if not data:
        break  # 没有更多数据，退出循环
    prs.extend(data)
    page += 1

print(f"Total PRs fetched: {len(prs)}")
```

PR 数据示例如下：

https://api.github.com/repos/yiisoft/yii2/pulls/10017

六、提交清单

提交材料应包括：

1. 完整的**实验报告**建议包含以下部分，可以适当调整结构：
    a. **任务一：预测 Pull Request 处理时间**

i. **问题与数据**：明确任务定义，说明所用仓库数量、数据来源、时间切分方式、防止数据泄漏的措施。

ii. **特征工程**：列出所有使用的特征及其计算方法。说明哪些特征在收集数据时可以获取，哪些需要额外查询；阐述清洗策略及缺失值处理方法。

iii. **模型与方法**：描述使用的模型，阐述参数选择理由。

iv. **结果与分析**：提供所有指标的可视化图表并进行结果分析。

v. **结论与建议**：总结发现，并提出对项目维护者或代码评审流程的建议。

b. 任务二：预测 Pull Request 结局

i. 同上。

c. 抓取数据说明

i. 描述获取数据对应的 API以及速率限制处理策略；

2. **源代码**、原始数据、处理后的特征数据等，可用 README 指导助教复现实验

作业提交截止日期：9 月 30 日

参考文献

[1] Y. Fan, X. Xia, D. Lo, and S. Li, Early prediction of merged code changes to prioritize reviewing tasks, Empirical Software Engineering, vol. 23, no. 6, pp. 3346—3393, 2018.

[2] M. I. Azeem, Q. Peng, and Q. Wang, Pull request prioritization algorithm based on acceptance and response probability, in Proceedings of the 20th International Conference on Software Quality, Reliability and Security. IEEE, 2020, pp. 231—242.

数据集

yii2.zip