homework12

任务要求

- 对GitHub上具有协作行为日志数据的500名用户的个人信息（包括姓名、公司、邮箱及其地理位置等）进行**数据洞察分析**。

数据获取

- 做了一半 发现数据很奇怪，于是乎想起群里赵学姐的提醒。下面代码就不删了，直接使用user_combined_infor_500.csv重命名为all.csv
- 数据获取->7个子文件
- 用python代码合并成一个**all.csv**文件保存在data/user_data文件夹下
- 代码如下

```
import os
import pandas as pd
# 指定文件夹路径
folder_path = 'data/user_data'
# 获取文件夹内所有的CSV文件
csv_files = [f for f in os.listdir(folder_path) if f.endswith('.csv')]
# 初始化一个空的DataFrame来存储所有数据
combined_data = pd.DataFrame()
# 循环读取每一个CSV文件并将其追加到combined_data中
for file in csv_files:
    file_path = os.path.join(folder_path, file)
    data = pd.read_csv(file_path)
    combined_data = pd.concat([combined_data, data], ignore_index=True)
# 将整合后的数据保存到新的CSV文件中
combined_data.to_csv('data/user_data/all.csv', index=False)
print('所有CSV文件已成功整合并保存为 "all.csv"')
```

实验目标

- 培养数据处理与分析能力：通过实际操作，提升对大规模数据集的处理和分析能力。
- 掌握GPT工具的应用：学习如何利用GPT大型模型工具辅助完成数据洞察任务。
- 理解数据隐私与伦理：在处理包含个人信息的数据时，遵循数据隐私保护的原则和规范。

实验内容

1. **人口统计分析**

- 国家和地区分布：统计用户所在国家和地区的分布，识别主要的开发者集中地。
- 城市级别分布：分析主要城市的开发者密度，发现技术热点区域。
- 时区分布:了解用户的时区分布，分析不同地区用户的协作时间模式。

2. **协作行为分析**

- 提交频率：统计每个用户的提交次数，识别高活跃用户和低活跃用户。

3. **其他维度有趣的洞察**（至少2个）

实现过程

0. 数据分析

- 首先要确认文件的字段
- 代码如下

```
import pandas as pd
user_data = pd.read_csv('data/user_data/all.csv')
print(user_data.head())
```

- 输出如下（转换成markdown表格的形式）：

user_id	name	location	total_influence	country	event_type	event_action
663432	bdraco	Houston, TX	1776.967163	United States	CreateEvent	added
663432	bdraco	Houston, TX	1776.967163	United States	CreateEvent	added
663432	bdraco	Houston, TX	1776.967163	United States	CreateEvent	added
663432	bdraco	Houston, TX	1776.967163	United States	CreateEvent	added
663432	bdraco	Houston, TX	1776.967163	United States	CreateEvent	added

- 有以下字段：`user_id, name, location, total_influence, country, event_type, event_action, event_time`

1. 人口统计分析

1.1 国家分布分析

- 根据数据中的`country`字段，我们可以统计不同国家的开发者数量，进而识别开发者的主要集中地。
- 代码如下

```
# 统计每个国家的用户数量
country_distribution = data['country'].value_counts()
# 输出前10个国家的用户数量
print(country_distribution.head(10))
```

- 输出如下：

```
country
United States     305788
Germany           182659
China              73011
United Kingdom     71606
France             59570
Canada             58600
Netherlands        52367
Czechia            48122
Japan              46553
Switzerland        38093
Name: count, dtype: int64
```

- **分析**：
- 我们可以看到开发者集中在少数几个国家，主要集中在：
  1. **United States** (305,788) – 远超其他国家，显然是技术和开发者的重镇。
  2. **Germany** (182,659) – 位居第二，欧洲的开发中心。
  3. **China** (73,011) – 作为全球人口最多的国家，排名第三，表明中国在全球开发者中也有重要地位。
  4. **United Kingdom** (71,606) – 排名第四，是欧洲的另一大开发者集聚地。
- **洞察：**
  - **美国**是全球最大的开发者社区，几乎是其他国家的两倍。
  - **德国**、**中国**和**英国**是排名前列的重要国家，表明这些地区有非常活跃的技术生态。

1.2 城市分布分析

- `location`字段包含了城市和国家信息，我们可以将其拆分并分析，提取出城市名进行统计。
- 代码如下：

```
# 提取城市信息
data['city'] = data['location'].str.split(',').str[0].str.strip()
# 统计每个城市的用户数量
city_distribution = data['city'].value_counts()
# 输出前10个城市
print(city_distribution.head(10))
```

- 输出如下：

```
city
Germany          111786
Prague            39461
San Francisco     27542
Japan             26986
Berlin            25978
New York          24334
Paris             19516
Palo Alto         19215
London            18400
UK                17789
Name: count, dtype: int64
```

- **分析**
- 城市分布数据显示，全球开发者在某些主要城市有较高的集中度：
  - **Germany** (111,786) - 这可能指的是**德国**这个国家的开发者，但也可能包括德国多个城市的开发者。
  - **Prague** (39,461) - 可能代表了捷克共和国的技术聚集地，显示出布拉格作为科技创新和开发中心的地位。
  - **San Francisco** (27,542) - 作为全球技术创新中心，硅谷的影响力依然显著。
  - **Berlin** (25,978) - 德国的另一大技术中心，成为技术人才的聚集地。
  - **New York** (24,334) - 美国的技术和金融中心，开发者数量也非常可观。
- **洞察：**
  - **德国的影响力最大**，无论是整体的国家级分布还是单个城市（如**柏林**和**布拉格**）。

- **旧金山**依然在全球技术开发者中占有重要地位，尽管其他城市也有不少活跃开发者。
- **巴黎**和**伦敦**作为全球技术中心，虽然排名靠前，但与**旧金山**相比，开发者的数量差距较大。

2. 协作行为分析

2.1 事件类型分析

- 分析不同`event_type`的分布，了解不同类型的事件在数据中的占比。
- 代码如下：

```
# 统计不同事件类型的数量
event_type_distribution = data['event_type'].value_counts()
# 输出前10个事件类型
print(event_type_distribution.head(10))
```

- 输出如下：

```
event_type
PushEvent                     410955
PullRequestEvent              201128
IssueCommentEvent             174806
PullRequestReviewEvent        151843
CreateEvent                   104371
DeleteEvent                    96999
PullRequestReviewCommentEvent  86198
IssuesEvent                    51205
ReleaseEvent                    9455
WatchEvent                      3809
Name: count, dtype: int64
```

- **分析**
- 输出数据显示了不同事件类型的分布，其中**PushEvent**（代码推送事件）占据了绝对主导地位：
  - **PushEvent** (410,955) – 代码推送事件最多，表明这是开发者活动中的主要形式，开发者最常进行的操作是提交代码。
  - **PullRequestEvent** (201,128) – 拉取请求事件也非常重要，说明很多开发者参与了代码审阅和合并的工作。

    - **IssueCommentEvent** (174,806) – 这个事件表明开发者参与了问题讨论和评论，显示出技术社区的协作性和互动性。
    - **PullRequestReviewEvent** (151,843) – 拉取请求的审查行为也非常普遍，说明代码评审是开发过程中不可或缺的一部分。
    - **CreateEvent** (104,371) – 创建事件可能代表了项目或仓库的创建，是技术协作的重要起点。
- **洞察：**
    - **PushEvent**的主导地位进一步强调了开发者在GitHub上的主要活动是代码的提交和更新。
    - **PullRequest**相关的事件（拉取请求和审查）说明了协作型开发的重要性，尤其是在大型项目 中。
    - **IssueCommentEvent**表明开发者对项目中的问题和讨论非常积极，体现了良好的社区互动。

2.2 事件行为分析

- 根据`event_action`字段，分析用户在每种事件类型下的行为分布。
- 代码如下：

```
# 统计不同事件行为的数量
event_action_distribution = data['event_action'].value_counts()
# 输出前10个事件行为
print(event_action_distribution.head(10))
```

- 输出如下

```
event_action
added        617218
created      411961
closed       173489
opened        76406
published      9455
started        3809
reopened       2438
Name: count, dtype: int64
```

- **分析**
- 根据`event_action`字段的输出，我们可以看出事件的具体行为模式：
    - **added** (617,218) – 表明最常见的操作是“添加”某些内容（如文件、代码等）。

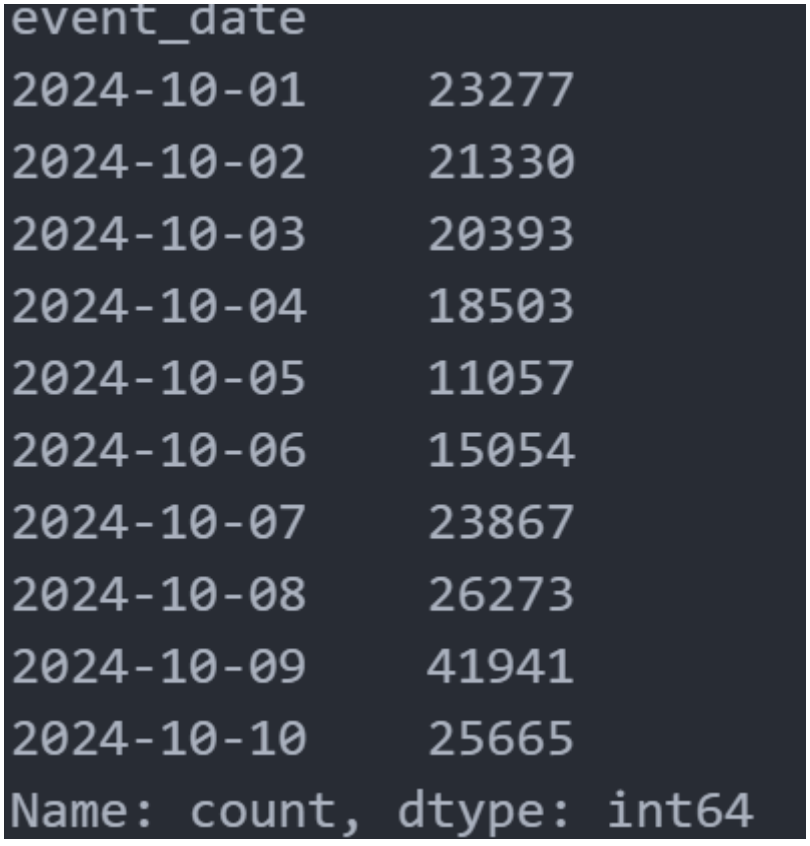
    - **created** (411,961) – “创建”操作也非常频繁，表明开发者不仅参与维护和改进现有项目，还积极发起新项目。
    - **closed** (173,489) – 关闭操作可能涉及问题、拉取请求或其他任务的解决。
    - **opened** (76,406) – 打开操作，通常是指新拉取请求或新问题的创建。
- **洞察：**
    - “添加”和“创建”是最常见的操作，反映出开发者主要集中在创作和扩展新功能或项目。
    - “关闭”和“打开”则指示了开发者在项目的生命周期中的管理和更新任务，显示出高活跃度和参与度。

2.3 事件时间分析

- `event_time`字段记录了事件发生的时间。我们可以根据日期对事件进行统计，分析用户活动的时间分布。
- 代码如下

```
# 将event_time转换为日期类型
data['event_time'] = pd.to_datetime(data['event_time'])
# 提取日期部分
data['event_date'] = data['event_time'].dt.date
# 统计每天的事件数量
daily_event_counts = data['event_date'].value_counts().sort_index()
# 输出前10天的事件数量
print(daily_event_counts.head(10))
```

- 输出如下



- **洞察：**
  - 从`event_date`的输出中可以看到，事件活动在**2024年10月1日**至**10月10日**期间的变化趋势。
  - 在10月9日，事件数量达到最高（41,941），这可能是某些技术发布或重要项目更新的高峰期。一般来说，这样的峰值往往发生在周中，可能与工作节奏和协作周期相关。
  - **10月5日到10月6日**之间的事件数量大幅下降，可能与周末的工作模式相关，显示出开发者活跃度受工作日影响较大。
  - 从整体数据来看，周初（如**10月1日和10月7日**）的活跃度较高，可能是因为新的工作任务开始或者开发者集中讨论新问题。

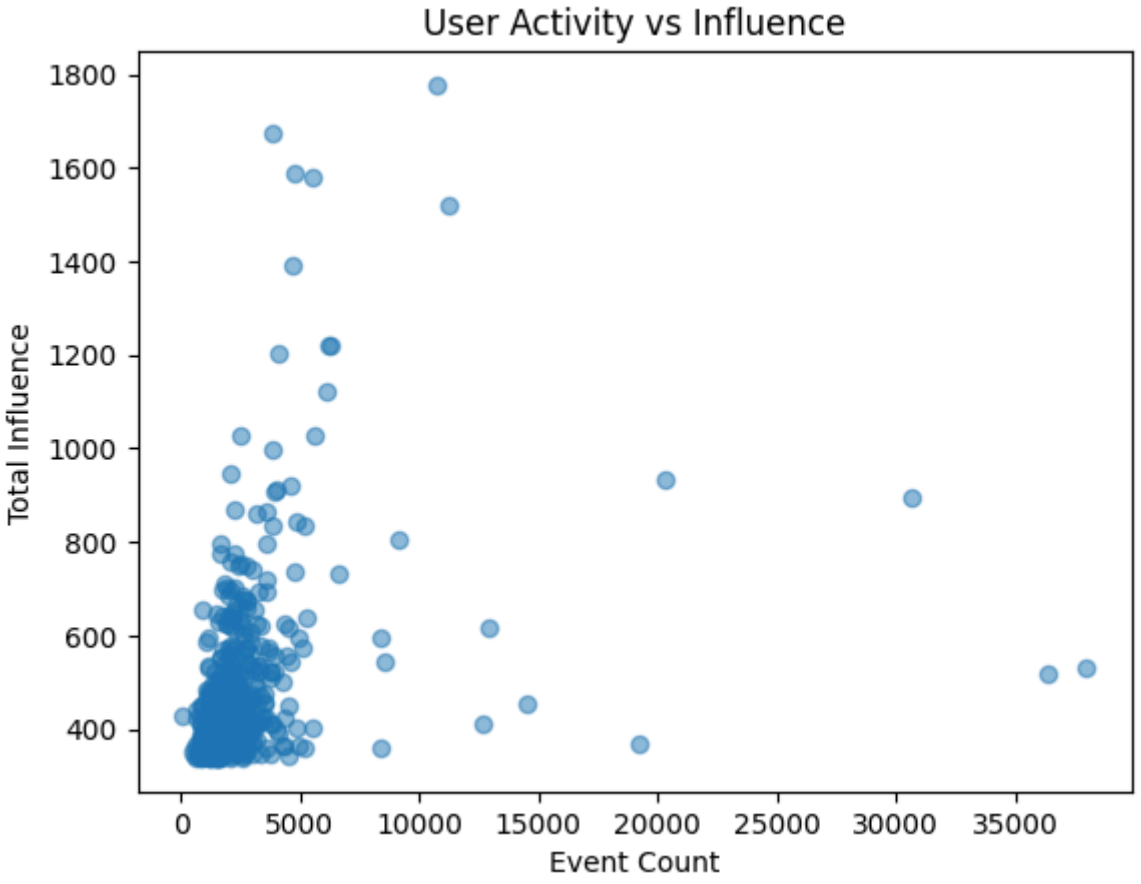
3. 其他维度的洞察

3.1 用户活跃度与影响力关系分析

- 根据`total_influence`字段，分析用户的影响力与其活跃度（假设根据`event_type`或`event_action`来衡量）之间的关系。
- 代码如下

```
# 计算每个用户的事件数量
user_event_counts = data.groupby('user_id')['event_type'].count()
# 将用户的事件数量和影响力合并
user_activity_influence = pd.DataFrame({'event_count': user_event_counts,
'total_influence': data.groupby('user_id')['total_influence'].first()})
# 绘制活跃度与影响力的关系图
import matplotlib.pyplot as plt
plt.scatter(user_activity_influence['event_count'],user_activity_influence['
total_influence'], alpha=0.5)
plt.title('User Activity vs Influence')
plt.xlabel('Event Count')
plt.ylabel('Total Influence')
plt.show()
```

- 输出如下



- **洞察**
  1. 大多数用户是“普通用户”
  2. 高活跃用户影响力波动较大

3.2 活跃事件类型与用户影响力的关系分析

通过分析活跃事件类型与用户影响力的关系，可以揭示哪些事件类型对用户的影响力有更大贡献。

- 代码如下

```
# 计算每种事件类型的影响力总和
event_type_influence = data.groupby('event_type')['total_influence'].sum()
# 输出每种事件类型的影响力
print(event_type_influence)
```

- 输出如下

```
event_type
CommitCommentEvent                 3.382054e+05
CreateEvent                        5.383339e+07
DeleteEvent                        5.351956e+07
ForkEvent                          1.138427e+06
GollumEvent                        3.041101e+05
IssueCommentEvent                  9.413180e+07
IssuesEvent                        2.707371e+07
MemberEvent                        1.990206e+05
PublicEvent                        2.470887e+04
PullRequestEvent                   1.143337e+08
PullRequestReviewCommentEvent      4.627761e+07
PullRequestReviewEvent             8.506086e+07
PushEvent                          2.210280e+08
ReleaseEvent                       5.222728e+06
WatchEvent                         1.984058e+06
Name: total_influence, dtype: float64
```

- **洞察：**
  - **PushEvent** (2.21亿) 的总影响力遥遥领先，表明代码提交（推送）仍然是GitHub上最常见的活动，并且是开发者协作中的核心行为。这类事件表明开发者频繁进行代码更新。
  - **PullRequestEvent** (1.14亿) 排在第二，紧随其后，说明拉取请求在开发过程中也占据重要地位。拉取请求不仅是贡献代码的一种方式，也是团队合作和代码审核的核心。
  - **IssueCommentEvent** (9,413,180) 和 **PullRequestReviewEvent** (8,506,086) 也显示出开发者在项目中的互动频繁，特别是通过评论和审查拉取请求来进行协作和反馈。
  - **CreateEvent** 和 **DeleteEvent** 的总影响力也相当高（分别为 5,383万 和 5,352万），意味着开发者不仅创建新项目，也会频繁地删除或清理旧项目或代码。