

数据洞察分析报告

姓名：肖璟仪 学号：10235501452

一、实验背景与目标

背景：GitHub 作为开源社区平台，汇聚全球开发者，其用户行为数据对理解社区结构、预测技术趋势至关重要。

目标：

- 培养数据处理与分析能力，提升处理大规模数据集的准确性和效率。
- 掌握 GPT 等大型模型工具在数据洞察中的应用，提升智能化水平。
- 理解并遵守数据隐私保护原则，确保在保护用户隐私的前提下进行数据分析。

二、数据收集与预处理

```
import pandas as pd
```

```
# 读取CSV文件
```

```
file_path = 'D:/users_combined_info_500.csv'
```

```
df = pd.read_csv(file_path)
```

对这个文件进行预处理，可以从哪些角度考虑？

对CSV文件进行预处理是一个多步骤的过程，旨在确保数据的准确性、一致性和可用性，以便进行后续的数据分析或机器学习任务。以下是对你提供的CSV文件（D:/users_combined_info_500.csv）进行预处理时可以考虑的一些角度：

1. 数据清洗：

- 处理缺失值：**检查数据中的缺失值，并决定是删除包含缺失值的行/列，还是用特定的值（如0、平均值、中位数、众数等）填充这些缺失值。
- 去除重复值：**检查并删除数据中的重复行，以避免在后续分析中出现偏差。
- 纠正错误数据：**检查数据中的明显错误（如不合理的数值、拼写错误等），并进行纠正。

2. 数据类型转换：

- 确保数据中的每一列都具有正确的数据类型（如整数、浮点数、字符串、日期等）。
- 对于日期和时间数据，可能需要将其转换为 `datetime` 类型，以便进行时间相关的操作和分析。

3. 数据标准化和归一化：

发现文件中一个用户的活动占了多行，因此有以下操作：

```
# 去除重复值
# 使用drop_duplicates()方法去除重复的行
df_clean = df_no_na.drop_duplicates() 这个方法无用，因为文件中没有重复的行，请按照id名去重
```

如果你的数据集 `df_no_na` 中实际上没有重复的行，但是你想要根据特定的列（比如 `id` 列）来去除潜在的重复项，那么 `drop_duplicates()` 方法仍然是有用的，但你需要指定 `subset` 参数来告诉Pandas你希望基于哪些列来去重。

在你的情况下，你可以这样做：

python

```
1 # 假设df_no_na是你的数据集，并且它有一个名为'id'的列
2 # 使用drop_duplicates()方法，并指定subset参数为'id'来去重
3 df_clean = df_no_na.drop_duplicates(subset=['id'])
```

此外，发现 `location` 列中含有许多伪地名，利用 `aigc` 工具将其删除。



要删除CSV文件中包含特定伪地名的行，你可以使用Python的 `pandas` 库。以下是一个示例代码，它会读取 `D:/users_combined_info_500.csv` 文件，删除 `location` 列中包含 `fake_locations` 列表中任何地名的行，然后将结果保存回一个新的CSV文件（或者覆盖

Ulm

The Netherlands请你找出以上地区对应的时区，比如`utc-5`，`utc-8`，并把这些地区和对应的时区写成一个字典用于python代码，请不要省略任何一个地区，除了伪地区你可以标明，你是一个勤恳的人

当然，以下是一个勤恳的人为你准备的，包含所有给定地区及其对应时区的Python字典。请注意，有些地区可能具有多个时区（例如某些大国或具有特殊政治地位的区域），但在这里，我尽量为每个地区分配了一个最可能或最常用的时区。对于伪地区或不明确的地区，我已在注释中标明。

python

```
1 timezones = {
2     "Houston, TX": "UTC-6",
3     "The Netherlands, Helmond": "UTC+1",
4     "Switzerland": "UTC+1",
5     "Brooklyn, NY": "UTC-5",
6     "Enschede, The Netherlands": "UTC+1",
7     "Utrecht, The Netherlands": "UTC+1"
```

然后针对后面的题目，进行了不同种类的数据清洗与准备：

```
# 筛选掉包含伪地名的行
# 假设location列存在，并且包含地名的单元格是字符串类型
df_filtered = df_clean[~df_clean['location'].isin(fake_locations)]

# 保存结果到新文件（或者你可以覆盖原文件，但这里为了安全起见，保存为新文件）
df_filtered.to_csv('D:/users_combined_info_filtered.csv', index=False)

print("筛选完成，根据用户名去重、删除伪地区结果已保存到 D:/users_combined_info_filtered.csv")

✓ 1.4s
筛选完成，根据用户名去重、删除伪地区结果已保存到 D:/users\_combined\_info\_filtered.csv

# 筛选掉包含伪地名的行
# 假设location列存在，并且包含地名的单元格是字符串类型
df_filtered = df_clean[~df_clean['location'].isin(fake_locations)]

# 保存结果到新文件（或者你可以覆盖原文件，但这里为了安全起见，保存为新文件）
df_filtered.to_csv('D:/users_combined_info_2.csv', index=False)

print("筛选完成，没有根据用户名去重但删除伪地区结果已保存到 D:/users_combined_info_2.csv")

✓ 1.3s
筛选完成，没有根据用户名去重但删除伪地区结果已保存到 D:/users\_combined\_info\_2.csv
```

三、人口统计分析

```
国家分布前10名:
country
United States    111
Germany          36
China            30
Canada           22
France           19
United Kingdom   18
Netherlands      18
Australia        11
Poland            10
Italy             10
Name: count, dtype: int64
城市分布前10名:
city
Berlin           12
San Francisco     11
Amsterdam         8
Seattle           8
London            8
Paris             8
Beijing           6
New York          6
Tokyo             5
Zurich            5
Name: count, dtype: int64
```

从上述数据可以看出，美国的开发者数量仍然处于领先地位，远超其他国家。德国、中国和加拿大紧随其后，开发者数量也相对较多。这些国家不仅在经济和科技方面处于全球领先地位，同时也拥有较为完善的科技基础设施和人才储备。值得注意的是，法国和意大利也进入了前 10 名，显示了欧洲在科技领域的强劲实力。

柏林仍然是全球开发者密度最高的城市之一，与旧金山并列第二。阿姆斯特丹、西雅图、伦敦和巴黎等城市也拥有较高的开发者密度。值得注意的是，北京也进入了前 10 名，显示了中国在科技领域的快速发展和人才储备的增加。此外，东京和苏黎世作为亚洲和欧洲的重要城市，也吸引了大量的开发者前来工作和生活。

我已经得到#统计城市与时区对应情况

```
timezones = {
    "Houston, TX": "UTC-6",
    "The Netherlands, Helmond": "UTC+1",
    "Switzerland": "UTC+1",
    "Brooklyn, NY": "UTC-5",
    "Enschede, The Netherlands": "UTC+1",
    "Utrecht, The Netherlands": "UTC+1",
    "Boston": "UTC-5",
    "Philadelphia, PA": "UTC-5",
    "San Francisco, CA": "UTC-8",等等)这些数据，我要完成的任务是：了解用户的时区分布，
分析不同地区用户的协作时间模式。请告诉我接下来我要做什么以及这样做的原因
```

了解用户的时区分布并分析不同地区用户的协作时间模式是一个涉及数据分析和时间管理的重要任务。基于您已经得到的时区数据（`timezones` 字典），以下是一些建议的步骤以及这样做的原因：

1. 数据清洗和整理

步骤：

- 确保 `timezones` 字典中的城市/地区名称是唯一的，没有重复。
- 如果可能，将城市/地区名称标准化（例如，使用一致的命名格式，如“City, State/Country”或“Country, City”）。
- 创建一个包含用户ID和城市/地区映射的数据结构（例如，另一个字典或 DataFrame），以便将时区与用户关联起来。

原因：

文心大模型3.5

- 唯一且标准化的城市/地区名称有助于避免在后续分析中的混淆。
- 用户ID与城市/地区的映射是分析用户时区分布的基础。

2. 时区分布分析

步骤：

- 使用 `timezones` 字典创建一个 Pandas DataFrame，其中包含城市/地区和对应的时区。
- 对时区进行分组，并计算每个时区的用户数量。
- 可视化时区分布（例如，使用条形图或饼图）。

原因：

- 了解用户的时区分布有助于识别主要用户群体所在的位置。
- 可视化时区分布可以直观地展示用户分布的地理格局。

3. 协作时间模式分析

步骤：

- 收集用户活动数据（例如，登录时间、任务开始和结束时间等）。
- 将用户活动数据转换为相应的时区时间。
- 分析用户在不同时间段的活动模式（例如，工作日与周末、白天与夜晚的活动差异）。
- 识别跨时区协作的热点时间段和潜在的时间冲突。
- 可视化协作时间模式（例如，使用热力图或堆叠面积图）。

1.UTC+1 时区用户最为活跃：

UTC+1 时区的用户数量最多，达到了 140 人，远超过其他时区。这表明该时区内的用户活跃度较高，可能是欧洲和非洲部分地区的主要时区。

2.UTC-8 和 UTC+8 时区用户数量紧随其后：

UTC-8 时区（主要包括太平洋时间，如美国西海岸），UTC+8 时区（主要包括中国标准时间、马来西亚时间等）的用户数量也较多，分别为 58 人和 40 人。这两个时区覆盖了世界上人口众多的地区，因此用户数量也相对较多。

3.其他时区用户分布相对均匀：

除了上述几个主要时区外，其他时区的用户数量相对较少，但分布相对均匀。这表明我们的用户群体在全球范围内都有一定的分布。

4. 24 小时内的时区活跃度变化：

从提供的 24 小时时区活跃度变化表格中可以看出，不同时区在不同时间段的活跃度存在差异。例如，UTC+1 时区在多个时间段内都保持了较高的活跃度；而 UTC-8 时区则在晚上和凌晨时段较为活跃。这反映了不同地区用户的作息习惯和协作时间模式。

建议：

1. 优化协作时间：
 - 根据时区分布情况, 我们可以为不同地区的用户提供更加合适的协作时间建议。例如, 对于 UTC+1 时区的用户, 可以安排在欧洲工作时间的会议或活动; 对于 UTC-8 时区的用户, 则可以安排在美洲工作时间进行协作。
2. 提升用户体验：
 - 针对未知时区的数据, 我们可以优化用户注册流程或信息填写界面, 引导用户正确选择或填写时区信息。同时, 对于重复或异常的时区数据, 我们可以进行清洗和整理, 以提高数据的准确性和可靠性。
3. 关注偏远时区用户：
 - 虽然 UTC+13 和 UTC-3 等时区用户数量较少, 但我们也应关注这些用户的协作需求和时间模式。通过提供更加灵活和个性化的服务, 我们可以增强这些用户的归属感和满意度。

四、协作行为分析

	user_id	submission_count	activity_level
0	225	2885	High Activity
1	1945	1526	Low Activity
2	2621	796	Low Activity
3	4196	1983	Low Activity
4	9582	2258	High Activity
..
492	91018726	4098	High Activity
493	92015510	1866	Low Activity
494	95597335	2113	High Activity
495	100913391	3177	High Activity
496	112826355	1680	Low Activity

[497 rows x 3 columns]

根据活动级别 (activity_level) 将用户分为了高活跃用户 (High Activity) 和低活跃用户 (Low Activity)。

五、其他维度有趣的洞察：

文件的列名有user_id name location total_influence country event_type event_action event_time, 我想要进行其他角度的洞察, 请你给出几个点

基于您提供的文件列名 (user_id , name , location , total_influence , country , event_type , event_action , event_time) , 以下是从不同角度进行数据分析的一些建议洞察点:

1. 地理位置分析:

- **用户地域分布:** 统计每个国家或地区 (country 和 location) 的用户数量, 以了解用户主要分布在哪里。
- **活跃地域:** 根据 event_time 统计不同地域的活跃时段, 分析用户在不同时间段的活跃度。

2. 用户影响力分析:

- **影响力分布:** 统计不同 total_influence 值的用户数量, 了解用户影响力的分布情况。
- **高影响力用户行为:** 分析 total_influence 较高的用户的行为模式 (如 event_type 和 event_action) , 看是否有特定的行为模式与高影响力相关。

3. 事件类型分析:

- **事件类型分布:** 统计不同 event_type 的数量, 了解哪些类型的事件更受欢迎或更常见。

我选择了:

1.用户影响力分析:

影响力分布: 统计不同 total_influence 值的用户数量, 了解用户影响力的分布情况。

2.事件类型分析:

事件类型分布: 统计不同 event_type 的数量, 了解哪些类型的事件更受欢迎或更常见。

1. 用户影响力分布特征

- **分布范围:** 用户的总影响力值分布在 400 到 1800 的区间内, 表明用户之间的影响力存在显著的差异。
- **集中趋势:** 从直方图可以推测, 用户影响力的分布可能呈现某种程度的偏态, 即大多数用户的影响力可能集中在较低的数值范围内, 而少数用户则拥有较高的影响力。这种分布模式在社交媒体、在线社区等平台中较为常见, 通常被称为“幂律分布”或“长尾分布”。
- **用户数量:** 每个影响力区间内的用户数量可能有所不同, 具体取决于影响力的分布密度。从直方图可以看出, 随着影响力的增加, 用户数量可能逐渐减少, 表明高影响力用户是少数。

2.

最常见的事件类型是 CreateEvent, 共有 317 个。

- IssueCommentEvent 和 CommitCommentEvent 分别有 21 个和 19 个, 位列其后。
- DeleteEvent 有 16 个, 也是一个相对常见的事件类型。
- PullRequestEvent、IssuesEvent、ForkEvent、GollumEvent 等类型数量较少。

六、结论与建议

1. **优化协作策略**：根据时区分布情况，GitHub 可以为不同地区的用户提供更加合适的协作时间建议，以促进全球用户的交流与协作。同时，可以开发智能推荐系统，根据用户的行为习惯和时区信息，推荐适合的合作伙伴和项目。

2. **提升用户影响力**：针对高影响力用户，GitHub 可以给予更多的关注和支持，如提供专属的社区管理权限、参与重要项目的机会等，以进一步激发他们的积极性和创造力。同时，可以通过举办线上或线下活动，如技术分享会、开发者大会等，为普通用户提供更多展示自己才华和提升影响力的机会。

3. **加强事件类型引导**：虽然 CreateEvent 是 GitHub 社区中的一项重要活动，但其他类型的事件如 IssueCommentEvent、CommitCommentEvent 等也具有一定的价值和意义。因此，GitHub 可以通过优化界面设计、提供事件类型引导等方式，鼓励用户参与更多类型的事件，以促进社区的多样性和活跃度。

4. **完善数据隐私保护**：在数据分析过程中，GitHub 应始终遵守数据隐私保护原则，确保用户数据的安全和合规性。同时，可以加强用户隐私设置和权限管理，让用户更加放心地使用 GitHub 平台。