时间序列分析项目实践

南京大学《用 Python 玩转数据》MOOC 课程

1. 实践背景

   如果你是 MOOC 网的数据分析师，现在有同学们每天学习的行为日志数据，你的老板希望你从中对用户的行为习惯做一个分析，构建用户画像。你拿到这份数据之后，开始观察…

2. 数据概况

   a) 你拿到的这份数据示例如下：

用户 id	时间戳	用户点击的课程 id
A	2020-0101-21:30:12	1001
A	2020-0101-21:31:16	1002
A	2020-0602-08:20:31	1003
B	2020-1001-22:20:31	1002
B	2020-1001-23:20:31	1003
…	…	…

   b) 其中每一列的含义是：

      i. 用户 id：这个用户注册时唯一的编号

      ii. 时间戳：时间格式如上，依循“年-月日-时:分:秒”

      iii. 用户点击课程 id：MOOC 上课程的唯一 id

   c) 注意

      i. 假设数据存储为标准的 csv

      ii. 记录含义：在时间戳这个时刻，当前用户点击了这个课程 id；

      iii. 假设这份数据已按照（用户 id,时间戳）排序；

3. 分析任务

   a) 基本分析：描述性统计

      i. 将时间戳转换为可处理的时间格式；

      ii. 统计每小时的用户点击总次数；

      iii. 统计 2020 年 1 月到 2020 年 6 月，每个月的用户点击总次数；

      iv. 统计每门课程一天内按小时的点击次数分布；

   b) 进阶分析：

   例如行为模式分析，如生成点击时间间隔列：计算此用户当前点击距离上次点击的时间间隔（以秒为单位），每个用户的首条记录时间间隔填充为 0s，有兴趣的同学可尝试自行完成。

4. 基于 pandas/numpy 实现的参考处理策略

```
# 首先从文件中读取 csv 数据表（假设如此）
data = pd.read_csv("user_log.csv")
```

   a) 将时间戳转换为可处理的时间格式；首先根据文档将目前不规范的时间进行格式化 https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

   分析 2020-1001-23:20:31 具有这样的如下格式：%Y-%m%d-%H:%M:%S，符号含义可查以上文档。

```
# a) 将时间戳转换为可处理的时间格式
this_format = '%Y-%m%d-%H:%M:%S'
data['format_time'] = pd.to_datetime(data['时间戳'], format=this_format)
# 'format_time' 是一个新列，是标准 datetime 的结构
```

b) 统计每小时的用户点击总次数；

```
# 首先生成每个记录所处的小时，用于根据小时计数
# 一种是根据原始 "时间戳" 进行字符串分解，提取小时
# 本方法采用已经解析的 format_time
data['hour'] = data['format_time'].dt.hour
hour_count = data['hour'].value_counts()
# 计数每个小时出现的次数，其中有可能缺少某个小时
# 下面我们对计数结果按照 0 点-23 点进行排列，缺少的小时次数填充为 0
sorted_hour_count = hour_count.reindex(range(0,24)).fillna(0)
print(sorted_hour_count)
```

c) 统计 2020 年 1 月到 2020 年 6 月，每个月的用户点击总次数；

```
# 首先选取 2020 年 1-6 月的数据，利用布尔向量
year_range = data['format_time'].dt.year == 2020
month_range = (data['format_time'].dt.month >= 1) & (data['format_time'].dt.month <= 6)
# 两个条件共同满足
data_valid = data.loc[year_range & month_range]
# 与上个任务类似，对 month 分组计数
month1_6_count = data_valid['format_time'].dt.month.value_counts()
sorted_month1_6_count = month16_count.reindex(range(1,7)).fillna(0)
print(sorted_month16_count)
```

d) 统计每门课程一天内按小时的点击次数分布；

```
data['hour'] = data['format_time'].dt.hour
course_hour_count = data.groupby(['用户点击的课程 id','hour']).count()
print(course_hour_count)
```

5. 小结

a) 时间序列操作要理解 datetime 数据结构，掌握部分用法，具体操作可以通过官方文档查询来确定；

b) timedelta 是 datetime 数据结构之间运算的结果（比如相减）

c) demo 数据上的整体流程

```
# 首先从文件中读取 csv 数据表（假设如此）
# data = pd.read_csv("user_log.csv")

df = '''用户 id,时间戳,用户点击的课程 id
A,2020-0101-21:30:12,1001
A,2020-0101-21:31:16,1002
```

```
A,2020-0102-08:20:31,1003
B,2020-0622-22:20:00,1002
B,2020-1001-22:20:32,1002
B,2020-1001-22:22:31,1002
B,2020-1001-22:22:32,1003
c,2020-1001-23:20:31,1001'''.split("\n")
columns = df[0].split(',')
df = [d.split(',') for d in df[1:]]

data = pd.DataFrame(df, columns=columns)


# a) 将时间戳转换为可处理的时间格式
this_format = '%Y-%m%d-%H:%M:%S'
data['format_time'] = pd.to_datetime(data['时间戳'], format=this_format)
# 'format_time' 是一个新列，是标准 datetime 的结构

# b) 统计每小时的用户点击总次数
# 首先生成每个记录所处的小时，用于根据小时计数
# 一种是根据原始 "时间戳" 进行字符串分解，提取小时
# 本方法采用已经解析的 format_time
data['hour'] = data['format_time'].dt.hour
hour_count = data['hour'].value_counts()
# 计数每个小时出现的次数，其中有可能缺少某个小时
# 下面我们对计数结果按照 0 点-23 点进行排列，缺少的小时次数填充为 0
sorted_hour_count = hour_count.reindex(range(0,24)).fillna(0)
print(sorted_hour_count)

# c) 统计 2020 年 1 月到 2020 年 6 月，每个月的用户点击总次数;
# 首先选取 2020 年 1-6 月的数据，利用布尔向量
year_range = data['format_time'].dt.year == 2020
month_range = (data['format_time'].dt.month >= 1) & (data['format_time'].dt.month <= 6)
# 两个条件共同满足
data_valid = data.loc[year_range & month_range]
# 与上个任务类似，对 month 分组计数
month1_6_count = data_valid['format_time'].dt.month.value_counts()
sorted_month1_6_count = month1_6_count.reindex(range(1,7)).fillna(0)
print(sorted_month1_6_count)


# d) 统计每门课程一天内按小时的点击次数分布;
# 首先生成每条记录所在的小时
```

```
data['hour'] = data['format_time'].dt.hour
course_hour_count = data.groupby(['用户点击的课程 id','hour']).count()
print(course_hour_count)
```