数据科学大作业研究报告

自动评分“捉虫”

小　组　人　数：　1

报　告　人　姓　名：　陆志晗

学　　号：　181250096

联　系　方　式：　181250096@smail.nju.edu.cn

学　　院：　软件学院

二〇二〇年 七 月

目录

一、小组信息

人数：1

学号：181250096

姓名：陆志晗

邮箱：181250096@smail.nju.edu.cn

Python 完成题目数量：28

分工职责：个人作业，单独完成本次大作业包括代码编写、功能实现和数据分析、报告撰写的全部工作

二、研究问题

1.背景与猜想

根据过往经验和个人体验，猜测同学们在做题时可能会为了通过测试取得分数而采用所谓“面向用例”式的编程方法。“面向用例编程”，是软件学院本科生惯用的黑话，是一种适用于如 LeetCode、Python 基础编程练习这样的、通过有限个测试用例、由系统自动评分的编程练习的编程方法，其主要思路是通过多次提交得到的结果反馈，尝试逆向推测出后台测试用例的内容，进而在代码文件中使用大量 if...else 条件判断语句有针对性地通过测试用例，是一种投机取巧的得分手段。除此之外，经由同学反馈和实际调查，还发现同学做答中存在未按课程要求使用 Python 语言，但仍通过测试获得满分的情况。以上两种做法不但无益于同学们的 Python 编程能力提升，无法达到命题与做题的目的，滋长消极怠工的坏习惯，而且在自动评分中很难被发现，直接降低了系统自动评分的可信度和参考价值，如果教师直接采用其给学生赋分，则会带来不公平，而完全排除系统全部由人工评阅又显然是不显示的，面临两难局面。

2.选题与目标

基于以上考虑，本项目期望通过数据分析，运用 Python 语言实现以下功能：

1）分辨学生作答中的面向用例编程、非 Python 代码与有效代码，形成相应的字典数据表；

2）剔除学生作答中的无效代码，仅保留有效的 Python 代码，为后续人工查验完成预处理筛选工作；

3）基于筛查后的有效作答重新进行基本统计分析，为先前助教在课上列出的相关统计数据纠偏。

项目主题定为：自动评分“捉虫”

```python
# 图结构/Misaka Network与测试_15801
n,m=map(int,input().split())
if n==2 and m==3:
    print(2)
elif n==1 and m ==500:
    print(163)
elif n==1 and m ==1000:
    print(362)
elif n==300 and m==300:
    print(29986)
elif n==1 and m==200:
    print(70)
elif n==6 and m==6:
    print(9)
elif n==2 and m==50:
    print(51)
else:
    print(31)

```

图 1 “面向用例编程”

```cpp
# 树结构/可怜的狗狗_1580799890800/user_id_60625
#include<cstdio>
#include<algorithm>
using namespace std;
const int maxn=3e5+10;
int n,k,sz,ts;
int a,b,c;
int s[maxn],hs[maxn],tt[maxn];
struct tree{int s,mid,lp,rp;}t[maxn*25];
void build(int l,int r,int k){
    t[k].mid=l+r>>1;
    if(l==r) return;
    t[k].lp=++ts,t[k].rp=++ts;
    build(l,t[k].mid,t[k].lp);
    build(t[k].mid+1,r,t[k].rp);
}
void put(int l,int r,int k,int nk,int p){
    t[nk]=(tree){t[k].s+1,t[k].mid};
    if(l==r) return;
    if(p<=t[nk].mid){
        t[nk].lp=++ts,t[nk].rp=t[k].rp;
        put(l,t[nk].mid,t[k].lp,t[nk].lp,p);
    }
    else{
        t[nk].rp=++ts,t[nk].lp=t[k].lp;
```

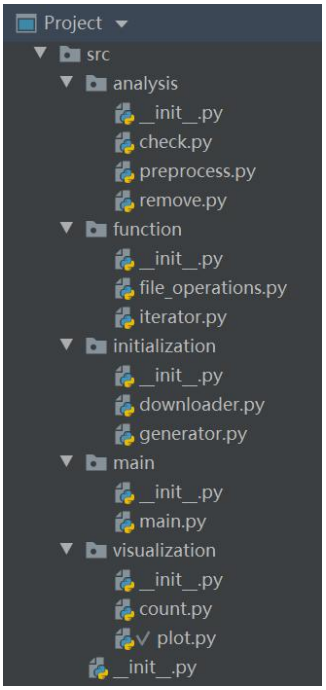
图 2 非 Python 代码

三、代码开源地址

1.地址

https://github.com/LoryKevin/Data-Science.git

2.项目结构



- 根目录 Data-Science
  - 源文件夹 src
    - 初始化文件夹 initialization：

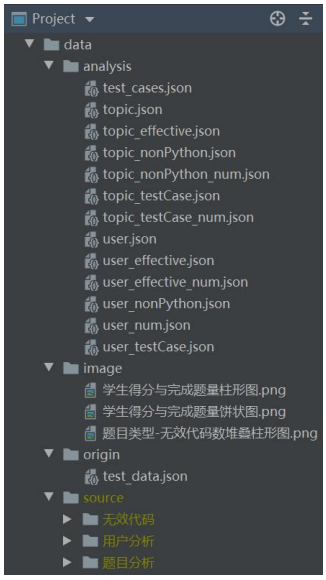
      用于初始化目录及下载源数据
    - 功能文件夹 function：

      用于提供迭代器和文件处理工具
    - 分析文件夹 analysis：

      用于分析源数据并将结果放到分析数据文件夹中
    - 制图文件夹 visualization：

      用于根据分析数据绘制统计图
    - 主文件夹 main：

      启动程序



- 数据文件夹 data
  - 原始数据 origin：

    助教提供的 test_data.json
  - 源数据：

    由原始数据下载得到的学生作答数据

    注：由于目录过多文件过大，未上传至仓库
  - 分析数据 analysis：

    分析中间数据和结果数据，主要是 json 字典文件
  - 图像数据：

    由分析数据绘制的统计图
- 文档文件夹 doc

图 3 项目结构目录

3.实现逻辑

开始分析后，main 函数首先调用 src/initialization/downloader 提供的 data_initialize()方法，生成文件夹目录，下载所需数据。之后，调用 src/analysis/preprocess 中 data_preprocess()方法下载数据预处理，删除损坏的文件夹，并生成两个字典文件 user.json 和 topic.json，分别对应用户为单位和题目为单位的目录。完成以上工作后进入分析阶段，src/analysis/check 的 code_check()方法包含面向用例编程和非 Python 代码两项检查，检查过程中生成两项检查结果两种排列方式共四个字典文件，这些字典文件在 src/analysis/remove 中被使用，将无效代码从源数据中移除，并更新 user.json 和 topic.json 生成 user_effective.json 和 topic_effective.json。最后，由 src/visualization 中 get_nums()和 graph_generate()两方法配合进行计数和绘图操作。

```
if __name__ == '__main__':
    print("分析开始！")
    data_initialize() # 数据初始化，包括文件系统初始化、下载源数据
    data_preprocess() # 数据预处理，包括检查数据下载情况、生成字典文件
    code_check() # 代码检查，包括面向用例编程和非Python代码检查
    code_remove() # 代码移除，将无效代码移至专门文件夹
    get_nums() # 频数计数，按题目或用户为单位计算多个频数
    graph_generate() # 图像生成，绘制统计图
    print("分析完成！")
```

图 4 main 函数调用

四、研究方法

本项目主要使用了描述性统计的数据分析方法。对调查总体所有变量，在本项目在即全体同学的全部作答情况，进行统计性描述，运用制表和分类、图形以及计筠概括性数据，将原有一系列复杂的数据集，如获得的用户分析、题目分析数据表用几个有代表性的数据进行描述，包括完成题数、无效代码数、使用面向用例编程频数、使用非 Python 代码频数等，进而能够直观地解释数据的变动，完成集中趋势分析、离散程度分析、频数分析、分布分析等统计工作。

由原始数据提取的字典文件中，具有如下字段（key）：

- user_id：用户 ID，不同学生用户注册时分配到的 ID，是用户的唯一识别码，每个用户 ID 不可重复
- topic_type：题目类型 ID，题目一级分类，表示题目属于哪个类别
- topic_id：题目编码，是题目的唯一识别码，每道题不可重复
- topic_num：题目数，表明某一分类下题目总数
- answer_num：作答数，表明某一分类下作答总数

分别以 user_id 和 topic_id 为关键字获得如下两个字典：

图 5 user 字典

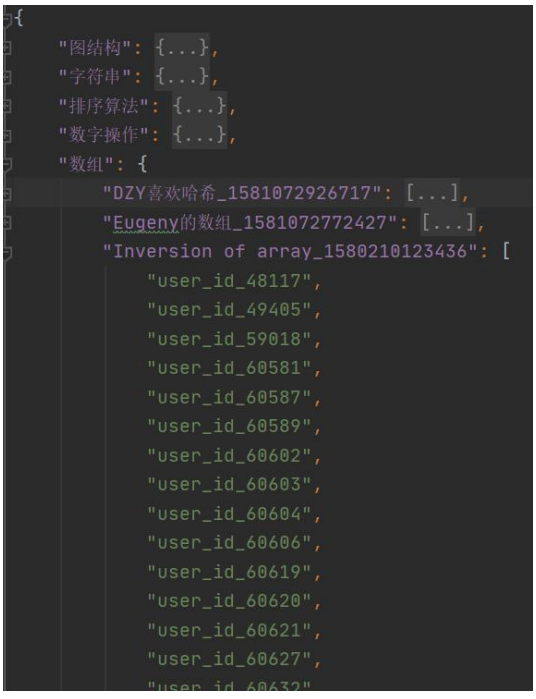
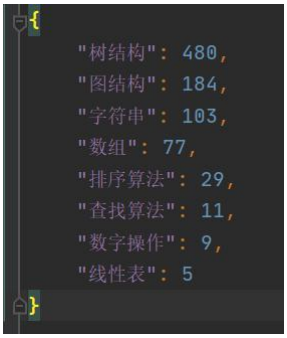


图 6 topic 字典

例如，在统计学生有效作答数分布时，则需要使用 user_id 和 topic_num 两个字段，生成分布图描述分布情况，用算数平均数 $M = \frac{X1 + X2 + \cdots Xn}{n}$ 、

中位数、众数描述集中趋势，用标准差 $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$ 描述离散程度。

而在分析使用面向用例编程及其他语言代码的频数和题目类型的关系时，则需要使用字段 topic_type 和 answer_num，列出个题目类型无效代码的分布情况，生成对应的统计图，通过频数分析反映各类型题目难易程度。



五、案例分析

1.修正后的学生得分/完成题量

在布置大作业的《数据科学基础大作业说明》PPT 中，助教统计了同学们 Python 联系完成情况，绘制了如下的学生得分/完成题量柱形图。由图中数据可以直观看到，众数落在“100 分/≥199 题”分段，换言之，绝大多数同学完成了全部的试题并拿到了满分，这似乎是自然且无需质疑的。但是经过上文所描述的分析过程，发现在这之中有相当一部分的提交存在面向用例编程和使用非 Python 代码的问题，不应计入有效提交，无法获得分数。于是，本项目对剔除无效提交后的数据进行分析，生成修正后的学生得分/完成题量统计图表

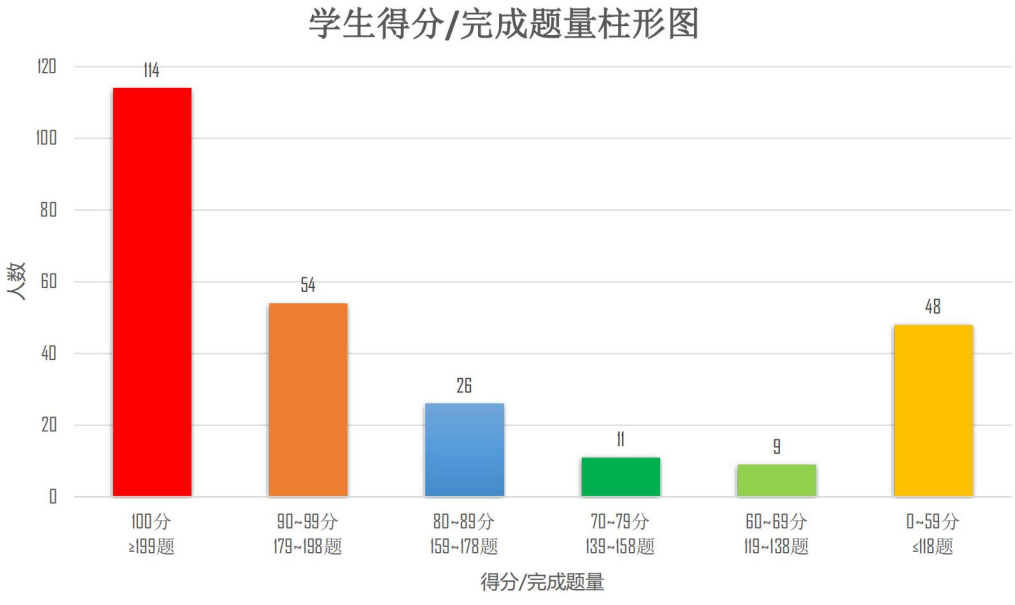
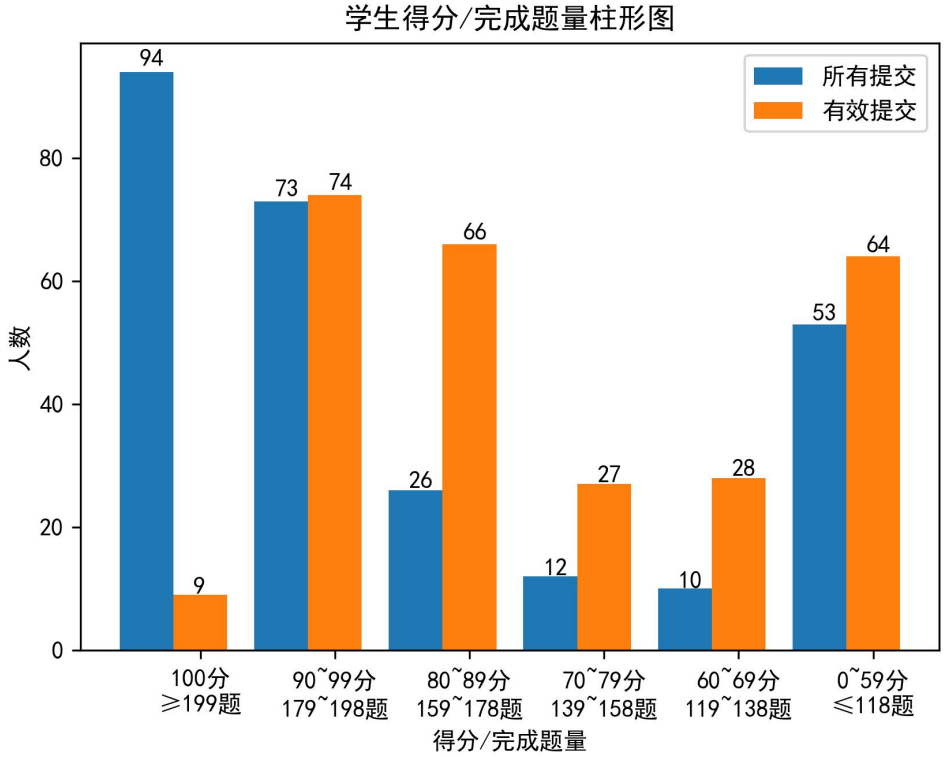


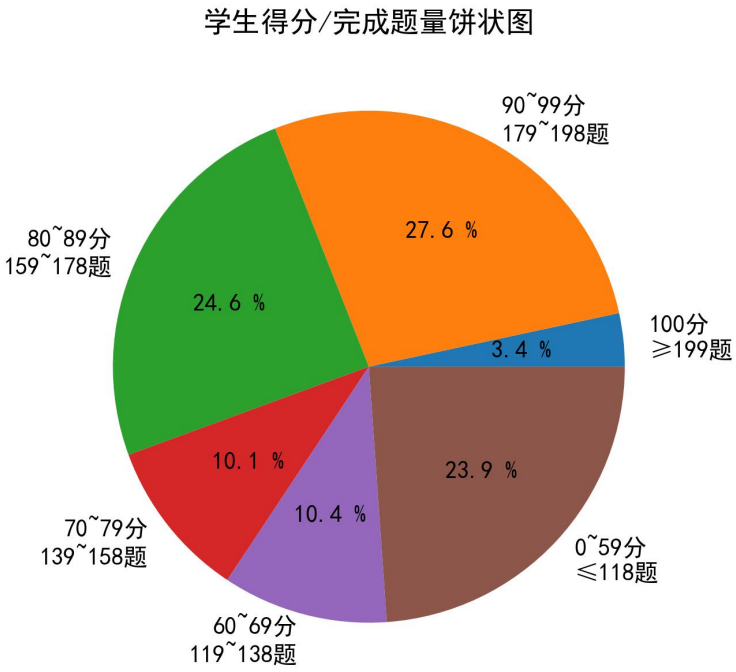
图 7 助教提供的学生得分/完成题量柱形图

在通过迭代器顺次取出所有提交进行两项检查后，生成了两份字典文件，分别列出面向用例编程的提交和非 Python 代码的提交。依据这两个字典剔除掉无效提交，仅保留有效的代码以备后续人工查验。随后，对于所有有效的代码重新生成一份 user_effective.json 字典，并对其进行计数，按由高到低的顺序列出各同学有效提交的题量，保存在 user_effective_num.json 文件中。使用 Matplotlib 数据绘图包绘制如下的柱形图和饼状图。


```
{
    "user_id_60793": 200,
    "user_id_60796": 200,
    "user_id_60598": 199,
    "user_id_60614": 199,
    "user_id_60621": 199,
    "user_id_60658": 199,
    "user_id_60716": 199,
    "user_id_60810": 199,
    "user_id_61132": 199,
    "user_id_60797": 198,
    "user_id_60593": 197,
    "user_id_60698": 197,
    "user_id_58586": 196,
    "user_id_60759": 196,
    "user_id_60791": 196,
```

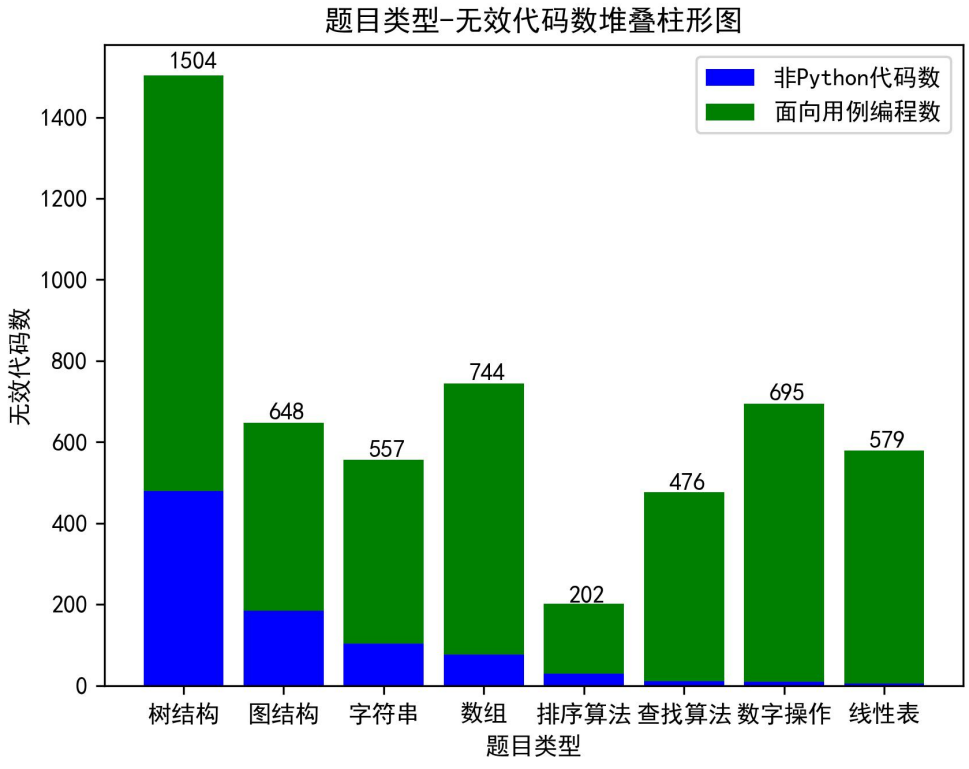



在这张并列柱状图中，蓝色柱显示了在剔除无效提交前所有提交的分布情况，橙色柱则是仅保留有效提交后完成题量的分布情况。可以看到，图形整体向右发生了偏移，满分的人数减少了超过 90%，仅有 9 人未使用面向用例编程或非 Python 代码，不折不扣地完成了几乎全部的题目。而其他分段的人数都有所增加，并多出了 11 人落到了 60 分以下。集中分布趋势在下方饼状图中展示得更为明显，可以看到，众数落在了“90~99 分/179~198 题”分段，经过更细致的计算，平均分数降到了 70.6，相较未修正时的 79.7 下降了近十分，可见无效代码对统计的影响显著。

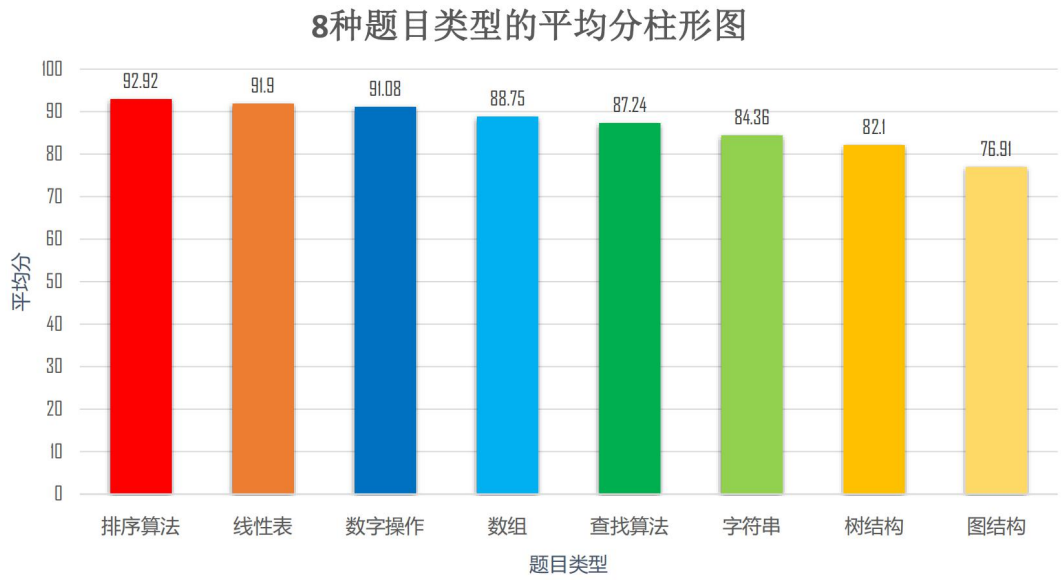


2.题目无效代码（面向用例编程及非 Python 代码）频数

同学使用面向用例编程或非 Python 代码，往往是由于不能正常正确地完成题目，而通过投机取巧的方法以获得分数，所以，无效代码的数量在一定程度上也能反映题目的难易程度。对此，本项目由希望能通过统计无效代码出现的频数从一个角度上初步反映各题目分类的难度，制作了如下的频数分布堆叠柱状图。



图中，蓝色柱展示了非 Python 代码频数，绿色柱展示面向用例编程数，并按非 Python 代码数由高到低自左向右排列。可以很明显地看到，无论是哪一中无效提交，树结构都是重灾区，而排序算法的无效提交是最少的，这也与助教提供的 8 种题目类型的平均分情况近似相符，可见确实可以初步反映题型难度。同时，图中也反映出，在各题型中两种无效代码出现比例并不总是相近，例如数字操作的非 Python 代码并不多，但面向用例编程数却来到了第二位，仅次于树结构。



六、总结与历程

由于个人在学期初的懈怠，甚至未能完成半数题目，导致无法组队，因而大作业整个过程进展得十分不顺利，前期策划考虑不周，选题过于简单也不敢尝试复杂的项目，开发过程中也遇到了很多困难，几近无法坚持完成。不过尽管如此，我认为老师这样的安排是合理的，未完成任务理应受到相应的责罚，现在大作业的艰难也就是在为学期初还债，咎由自取。在个人作业的过程中，基础能力的差距被进一步放大，对课程的掌握也欠缺很多，以致在数据分析层面使用到的方法较为简单和基础，甚至可以说是幼稚了。并且，由于选题和规划更侧重功能实现，着重在代码筛查而非分析，所以用到的数据分析方法也并不多。整个大作业，只能说是勉强完成任务了吧，特别是看到其他同学在朋友圈发出的，或详实具体的研究报告，或美观精致的功能页面，再和自己的相对比，完全没有大功告成的喜悦，只有对之前懒惰的遗恨。我想我在这门课，或是数据科学的学习道路上还有很长的路要走，希望这次充满遗憾的大作业能成为一种卧薪尝胆的鞭策。