## 总体思路

本方案主要分为两个部分：一是如何分离出A与C；二是用W、Q的并集与treatment对outcome建模（下文用WQ表示W与Q的并集）。

### 如何分离出A与C

在A.ipynb里利用卡方检验与斯皮尔曼相关性可以简单地分离出A，然后用d分离检验检测非A的变量集合，恰好检测出V\_16大概率属于W。

在C.ipynb里根据WQ与C被(treatment,outcome)d分离。可以将WQ与C分离出来。具体方案为：如果假设x、y是待考察的字段，如果在（treatment,outcome）的基础上，加上x能提升y预测的性能，则认为x与y属于同一集合（WQ或C）。通过C.ipynb里的代码逐个检验变量，最终可以将非A的变量分成两大集合，这两大集合中包含V\_16的是WQ，不包含的是C。

最终的变量分离结果记录在V.md文件中。

### 如何建模

模型的输入为WQ与treatment，输出为outcome，损失函数为均方误差，使用单层隐藏层全连接神经网络建模即可。其中有几个细节要注意一下：

1. 对于outcome需要做一下标准化处理，将其按照treatment分组，使其各组的均值为0。良好的输入输出处理有利于提升模型的性能。
2. 由于神经网络难以处理序信息。所以对于每个连续变量，都按大小计算出序号，作为额外连续特征加入神经网络中。
3. 对于每个输入的连续变量，需要将其归一化到0均值1方差。良好的输入输出处理有利于提升模型的性能。
4. 由于输入中存在Q集合，容易造成过拟合，所以需要在加上dropout缓解。

## 文件

* A.ipynb 用于分离出A
* C.ipynb 用于分离出C
* V.md 记录分离结果
* tools.py 工具函数模块
* train.py 训练与预测代码，执行后会在O文件夹分别针对3种treatment生成3个预测文件。
* stack.py 模型平均代码，用简单求均值即可
* run.sh 训练与预测入口，主要功能是调用train.py 6次，然后调用stack.py平均6次预测结果并输出result.csv
* run2.sh 功能与run.sh相同，如果执行run.sh遇到显卡显存不足的问题，请执行run2.sh。
* O 这是文件夹，储存训练的中间结果
* train.csv 主办方提供的文件
* test.csv 主办方提供的文件

## 环境要求

* xgboost gpu版本，用cpu版本也行，需要稍微改一下代码。
* torch== 1.8.2
* scipy
* sklearn
* pandas
* numpy
* matplotlib

## 额外的话

最后的建模方案是为了尽量取得高分而设计的，使用6个相同结构，不同初始化的神经网络训练取平均得到，使用6张3090训练20分钟可以训练完成。

如果为了快速出复现，训练1个神经网络模型即可，分值不会差太多，不影响排名。

另外如果用于复现的卡的数量大于等于6，请把train.py里的



这个地方的注释取消掉，用于提升训练速度。