刚开始的思路

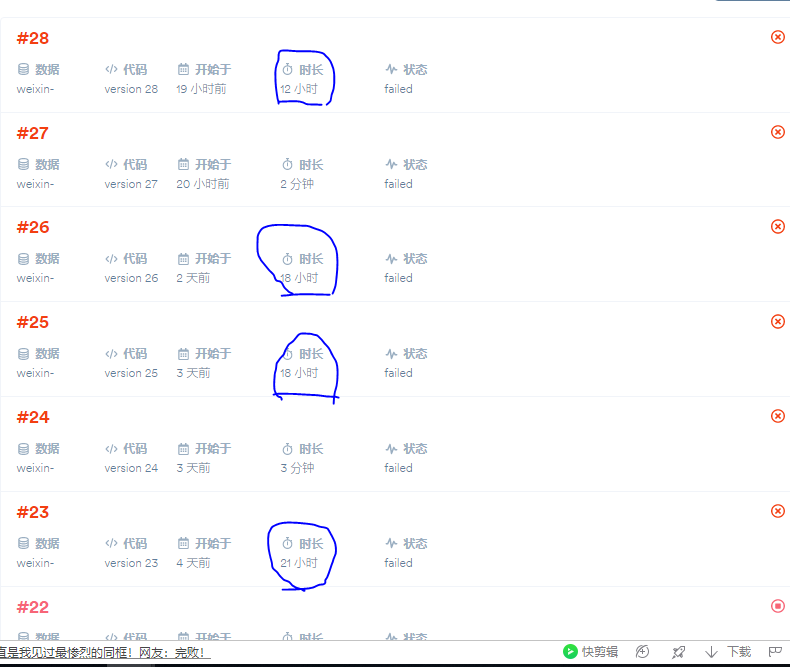
采用基于矩阵分解的协同过滤来做：

训练集中只需要train.csv：

1. 获取用户id-msno、歌曲id-song\_id、是否订阅-target
2. 计算用户对每一首歌曲的打分（根据用户id，对订阅次数求和得到总的订阅次数，**用每首歌曲的订阅除以总订阅次数**）
3. 生成user-item-scores用户-物品关系矩阵
4. 初始化隐含变量维数K=40，得到pu qi（R=pu\*qi）
5. 采用梯度下降法，训练bu bi pu qi，步数step=50（步数太长，后来调整为15）
6. 对于测试集，每一行的样本输入用户id-uid，歌曲id-iid

得到预测分数：**score=mu+bi[iid]+bu[uid]+np.sum(qi[iid]\*pu[uid])**

但是，在代码调试过程中，每次训练都要等待10几个小时，太漫长了。所以放弃了这个方法，转向LightGBM



目前思路：

采用LightGBM方式处理，属于二分类问题：

LightGBM能直接处理类别型特征，但是输入值为**int类型**

特征工程：

1. 将source\_screen\_name、source\_type、genre\_ids、composer、lyricist、bd、isrc中的作为类别型特征，采用类似lable\_Encoder()方法处理，给特征中出现的每一个类别赋予一个数值，作为类别型特征输入lgbm
2. gender采用get\_dummies()方法进行编码，类似One-hot，处理完作为类别型特征输入
3. song\_length 作为数值型特征直接输入
4. msno、song\_id不参与训练与测试

**优点**：将source\_screen\_name、source\_type、genre\_ids、composer、lyricist、bd、isrc这些特征通过map()方法映射为对应数值的方法，并没有增加特征的维度