离散工件制造文档

————huahua 队伍

电脑配置

- Cpu 为 i5-7300HQ
- 内存 16G
- 显卡：GTX1050Ti 2G

运行环境

语言：Python3.6.2
第三方库依赖：

- numpy
- pandas
- sklearn
- catboost
- lightgbm
- gc
- warnings

文件说明

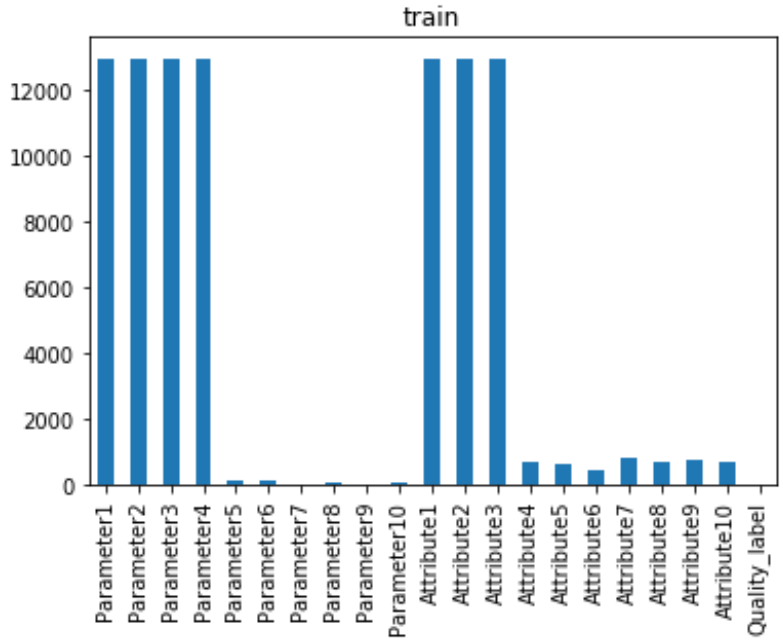
- data：存放训练数据和测试数据，文件命名和官方给的一样，train.csv 是由初赛和复赛训练数据合并而成。
- last / last_cbt / last_lgb：存放生成的 sub 文件。
- last_cbt.py / last_lgb.py / last_3B_cbt.py / last_3B_lgb.py：四个.py 文件均为训练文件，有四个训练文件，但是只是改变了模型和参数，还有最终特征，总体太多变化，用于最后的模型融合。
- stackEm.py：存放模型融合的代码，包括最大值融合和线性融合。

建模过程

1.数据分析

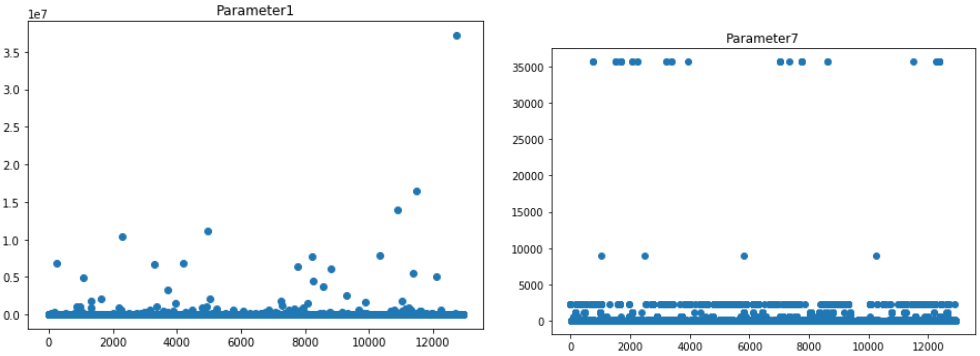
1.1 对训练数据进行分析：

在初赛中通过对数据的画图分析，可以发现 A 类 5-10 特征和 B 类 4-10 特征是离散的特征。图一可见：



图一

对原始数据特征进行画图分析，可以发现大部分数据为长尾发布，数值之间的极差很大，这对后续的特征工程会产生影响。图二可见：



图二

通过观察初赛和复赛的训练数据集，可以发现复赛训练集的前 6000 条数据和初赛训练集的 6000 条数据在离散特征上是一样的，但是在连续特征上是不同的。见图三：

	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	Attribute7	Attribute8	Attribute9	Attribute1	Quality_label
2	0.00166	0.591013	147.6084	38.18634	0.000421	0.000612	2286.523	0.035407	0.593081	1.010385	6.856075	0.168761	1.098755	36.95599	8.454598	11.43807	177.2431	338.7293	2.021704	0.079526	Pass
3	1.601749	0.015052	0.035864	51.13033	0.000909	0.002397	2286.523	0.035407	0.593081	1.010385	0.000362	11.64903	0.066671	225.6329	0.48186	20597.45	3.72333	15.37619	0.986973	4.634376	Fail
4	0.098039	69.23368	0.08092	0.112265	0.000909	0.001972	2286.523	0.035407	0.593081	1.010385	0.022201	0.078213	110.0797	2.208138	0.073525	236.0793	0.064196	0.576302	33.87579	1.813727	Fail
5	18.18186	0.047325	0.018061	1.098102	0.000909	0.002397	2286.523	0.035407	0.593081	1.010385	1.459004	0.380281	0.011491	0.654517	0.025872	176.9489	0.029777	0.246726	27.11717	0.081819	Fail
6	0.012085	0.008749	0.005509	524.3274	0.000909	0.002397	2286.523	0.035407	0.593081	1.010385	11.57665	1.555672	38.61339	0.260989	0.00938	194.798	0.055053	0.014725	13.56971	18.1385	Fail
7	0.004062	14.55648	0.786945	0.010545	0.000525	0.001623	2286.523	0.035407	0.593081	1.010385	0.001555	8.99894	6.392712	16.10748	1.016071	86.06426	0.57638	123.0575	16.13388	0.598517	Good
8	0.438449	1.232559	2.882699	0.610757	1.600654	0.464037	0.600827	17.85002	0.05185	0.010192	10.69064	0.034557	0.000971	1.021246	1.791292	0.377312	0.035493	0.14669	41.28538	5.985572	Good
9	48159.92	0.002987	14.86381	0.063287	1.43406	0.314162	0.600827	17.85002	0.05185	0.010192	0.057851	0.255094	0.090395	21.03551	29.60863	0.581454	0.615609	3.321156	0.013034	0.045039	Good

（1）

	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Parameter	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	Attribute7	Attribute8	Attribute9	Attribute1	Quality_label
2	0.167898	104.444	2.772825	0.146548	0.000421	0.000612	2286.523	0.035407	0.593081	1.010385	0.323881	2.59782	41.50648	36.95599	8.454598	11.43807	177.2431	338.7293	2.021704	0.079526	Pass
3	252.4831	0.343232	0.066873	0.002495	0.000909	0.002397	2286.523	0.035407	0.593081	1.010385	0.004594	0.004243	0.135967	225.6329	0.48186	20597.45	3.72333	15.37619	0.986973	4.634376	Fail
4	4.124654	0.170534	0.3838	4.27E-06	0.000909	0.001972	2286.523	0.035407	0.593081	1.010385	0.031295	0.951186	0.000423	2.208138	0.073525	236.0793	0.064196	0.576302	33.87579	1.813727	Fail
5	294.6567	6.153711	0.014716	4284.326	0.000909	0.002397	2286.523	0.035407	0.593081	1.010385	1.480634	0.000122	8.36E-05	0.654517	0.025872	176.9489	0.029777	0.246726	27.11717	0.081819	Fail
6	0.026284	0.16681	7.587398	0.002202	0.000909	0.002397	2286.523	0.035407	0.593081	1.010385	867.3423	0.827308	0.040846	0.260989	0.00938	194.798	0.055053	0.014725	13.56971	18.1385	Fail
7	0.049475	0.003962	1.060457	0.932693	0.000525	0.001623	2286.523	0.035407	0.593081	1.010385	3.644898	0.140683	101.541	16.10748	1.016071	86.06426	0.57638	123.0575	16.13388	0.598517	Good
8	0.100799	327.9809	0.397523	142.8132	1.600654	0.464037	0.600827	17.85002	0.05185	0.010192	0.001282	0.240269	0.645109	1.021246	1.791292	0.377312	0.035493	0.14669	41.28538	5.985572	Good
9	12.12031	0.001799	0.678944	227.7254	1.43406	0.314162	0.600827	17.85002	0.05185	0.010192	0.413215	0.350764	0.007765	21.03551	29.60863	0.581454	0.615609	3.321156	0.013034	0.045039	Good

（2）

图三、（1）为初赛训练数据，（2）为复赛训练数据

再对离散特征和 label 之间的关系进行分析，因为测试数据中没有给出 B 类特征，所以我们只进行 A 类特征的分析。对 A 类离散特征和 label 数据建表，去重；再去除 label，对 A 离散特征建表，去重；可以发现，行数减少了，所以可以看出离散特征相同的样本中有 label 不同的样本，说明连续特征应该是有识别作用的，和比赛期间大佬们说要去除 A 类特征 1-4 不太一样，本方案选择保留这些特征。见图四：

```
t.drop_duplicates().shape
(9544, 7)

t.iloc[:, :-1].drop_duplicates().shape
(7323, 6)
```

图四、上面为有 label 时的去重结果，
下面为无 label 时的去重结果。

1.2 对测试数据进行分析：

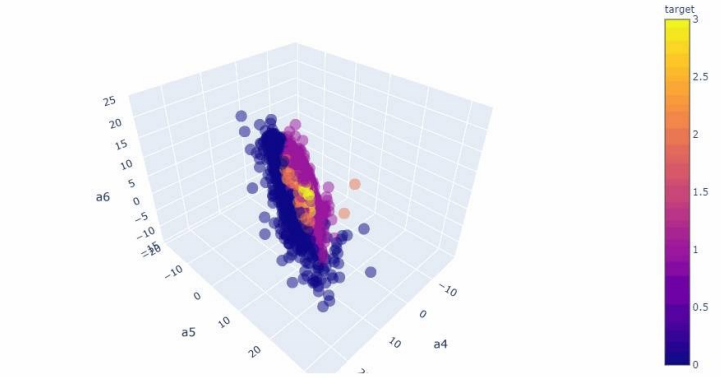
然后观察复赛测试数据，发现 Parameter9 特征中有一半的缺失值，后续会进行对缺少值的处理。见图五：

```
In [2]: test.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 11 columns):
Group          6000 non-null int64
Parameter1     6000 non-null float64
Parameter10    6000 non-null float64
Parameter2     6000 non-null float64
Parameter3     6000 non-null float64
Parameter4     6000 non-null float64
Parameter5     6000 non-null float64
Parameter6     6000 non-null float64
Parameter7     6000 non-null float64
Parameter8     6000 non-null float64
Parameter9     3000 non-null float64
dtypes: float64(10), int64(1)
memory usage: 515.7 KB
```

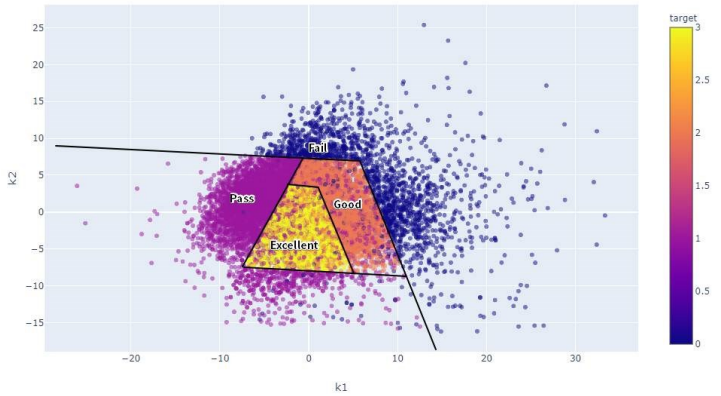
图五

1.3 对 B 类特征的分析：

在比赛后期对 B 类特征的分析中，学习了其他选手的想法，对 B 类 4-6 特征进行对画图观察，会发现这三个特征对 label 的区分比较明显。根据画图观察，特别是群中发出的图四（2），可以明显的看出，B 类 4-6 特征预测出的 k1 和 k2 的分类效果很明显。我受到启发，去除了 B 类的其他特征，并且优化了对 B 类 4-6 特征的预测，直接用三个特征作为特征进行了单模型的调整，用于后续融合。见图六：



（1）



（2）

图六、（1）是用 B 类 4-6 特征对 label 的画图观察，（2）是引用了群里的图片。

2.数据处理

根据上个阶段的分析，本方案在这个阶段做了如下处理：

- 合并初赛和复赛的训练集，根据上阶段对初赛和复赛训练数据的分析，可以发现初赛复赛的训练数据并不完全相同，合并数据集或许会让连续特征更好的发挥作用，并且合并数据集可以增大数据量，防止模型过拟合。
- 填充 Parameter9 特征，这里使用前一位数据进行填充，理由是通过对初赛测试集的数据观察发现，同一个值会连续出现数次，也就是说一个位置的值和它上下的位置的值有一定的关联。
- 根据上个阶段对原始数据的画图分析，可以看到，原始数据为长尾发布，不利于特征工程构造，所以对所以原始数据进行取对数操作（np.log10）。

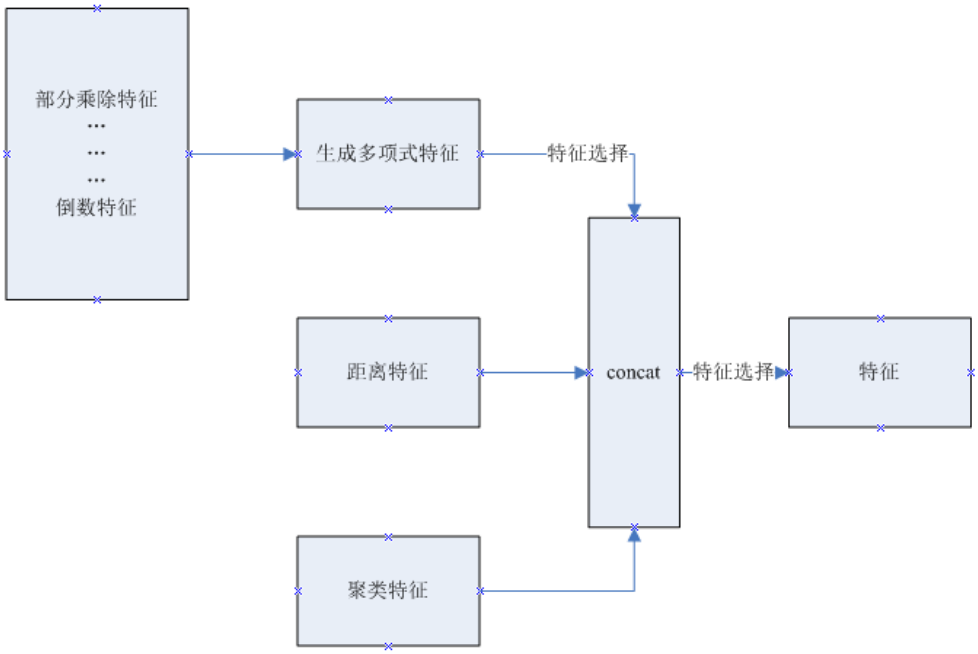
3.特征工程

本次比赛的原始特征均为匿名特征，无法了解每个特征的特殊含义，所以如果特征做的不是很多，并且使用了 sklearn 的多项式特征生成来构造有用特征。

以下是部分特征工程：

1. 多项式特征构造，用以下特征进行多项式特征构造，然后使用单变量特征选择，减少特征数量，得到第一部分特征：
   A、部分乘除特征
   B、离散数据的 count 特征
   C、原始离散数据连接进行 count 特征
   D、倒数特征
   E、原始特征
2. 每类 label 中每类原始数据减去此类数据平均值，意义为每类 label 中每类原始数据到此类数据中心的距离，此为第二部分特征。
3. 用聚类算法，对 A 类 1-10 特征进行聚类，聚类类别由手肘法确定，得到第三类特征。（但是效果似乎不太好，此类特征在最后的特征筛选中基本被去掉了）

最后将上述三类特征 concat 在一起，得到最终的 B 类数据训练特征。



图七、特征构造流程图

4.模型训练

本方案的训练思路是先进行 B 类 4-6 特征的预测，再进行 label 值的预测。在分析阶段中我们看到了 B 类 4-6 特征可以有效地进行 label 分类，所以我们在训练阶段就舍弃了其他 B 类特征，只使用三个 B 类特征。

1. 先预测 B 类 4-6 特征，然后和特征工程中得到的特征组合成最终特征，进行 label

预测。

- last_cbt.csv：使用 lightgbm 模型预测 B 类 4-6 特征，最后使用 catboost 预测 label。
- last_lgb.csv：使用 catboost 模型预测 B 类 4-6 特征，最后使用 lightgbm 预测 label

2. 先预测 B 类 4-6 特征，然后把预测出的 B 类 4-6 特征作为最后的特征进行 label 预测。
   - last_3B_cbt.csv：使用 lightgbm 模型预测 B 类 4-6 特征，最后使用 catboost 预测 label。
   - last_3B_lgb.csv：使用 catboost 模型预测 B 类 4-6 特征，最后使用 lightgbm 预测 label

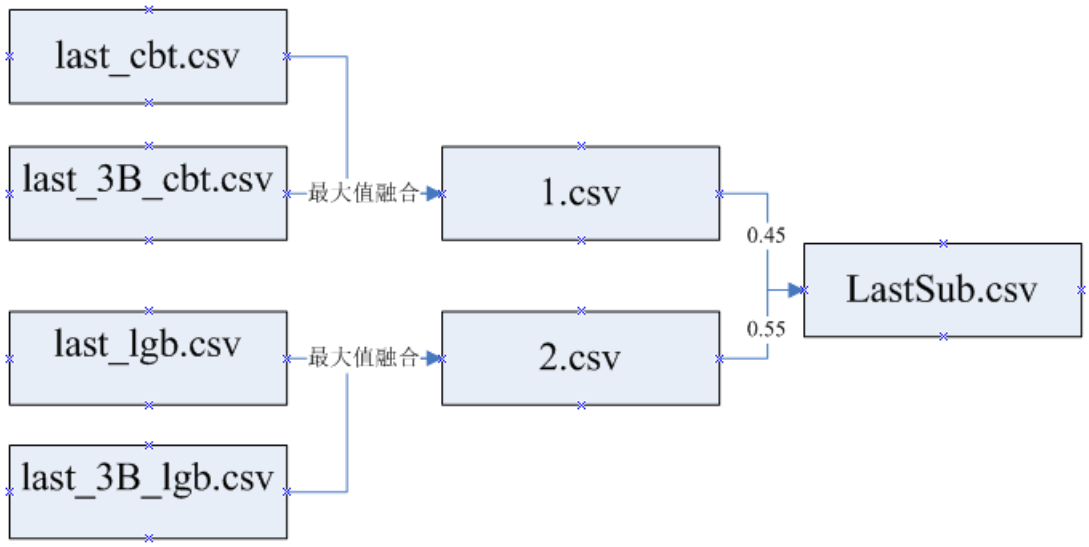
上面四个模型只在模型和参数以及最后的使用特征上有区别，其他部分都一样。根据上述的模型训练，现在得到了四个 sub，用于最后的模型融合阶段。

5.模型融合

本方案的模型融合中使用了两种模型融合方法，最大值融合和线性加权融合。下面简单说明一下这两种融合方法。

- 最大值融合：每个预测文件中，取出相同 Group 的组进行比较，把最大值最大的那个组作为最后的预测结果。这种方法的意义是使差异不是特别大的模型可以综合起来得到一个更加精确的结果，使这类型的模型在自己能准确预测的样本中得到更加精确的结果。从 sub 文件上看就是 last_cbt.csv 和 last_3B_cbt.csv、last_lgb.csv 和 last_3B_lgb.csv。
- 线性加权融合：这个比较常见，给每个 sub 文件分配一个系数，然后相加起来，减缓预测错误的样本带来的误差。

接下来是本方案的模型融合思路，首先是 last_cbt.csv 和 last_3B_cbt.csv、last_lgb.csv 和 last_3B_lgb.csv 分别进行最大值融合，得到两个 sub 文件在进行线性加权融合得到最终提交的 LastSub.csv 文件。



图八、模型融合

总结

本方案是在复核赛 b 榜自动评测完后结合前面的经验重新考虑而成的，所以只在 b 榜提交了，不太清楚 a 榜效果如何。因为是只有一次机会，本赛题抖动性又比较大，所以本方案使用了模型融合来增大模型的泛化能力，使用最大值融合希望尽可能得到同种模型下的最好结果，然后进行线性加权融合来提高泛化能力。

本方案流程比较常规，分析->处理->特征->训练->预测，方案比较普通，没有其他大佬的比较新奇的想法，也是按照 A->B->C 的顺序进行 label 预测，希望模型有更好的解释性。

根据对训练时的 loss 输出来看，本方案给出的模型有在进行学习，但是到后面的训练中 loss 下降慢，说明学习遇到了瓶颈，在特征工程方面或许还有改进的地方。并且参数方面也没有进行过多的调整，因为设备不是很好，训练比较慢，所以使用的是普遍使用的参数，对参数进行一下调整或许会更好些。

以上就是本方案的简单说明。