解决方案（简要说明）

作者：如梦落樱满熏香

1 赛题说明

西南财经大学“新网银行杯”数据科学竞赛给出的数据是：带标签数据 1.5*w* 条、不带标签数据 1*w* 条、预测数据 1*w* 条，数据集中存在较为严重的缺失。原始指标数有 157 个，其中 *x*1-*x*95 是数值型指标，而 *x*96-*x*157 是类别型指标。赛题设定的评价指标是不同客户群（客户群 1、客户群 2、客户群 3）的 *AUC* 加权平均值，权重分别为：0.3、0.3、0.4。

整个解决过程中本人只用了 *XGBoost*，其中存在特征筛选和迭代半监督。在解决问题的探索历程中，尝试了一些处理方法和模型，有些已经记不清楚了。但是有一些个人的心得体会和模型的理解和设定在文中会予以阐述，其中有些事尝试所得，有些是有明确的理论先验指导，但是有些就是经验和直觉了。总的来说，大道至简。

本文说明：*x*1 指的是带标签数据集中的自变量数据集、*x*2 指的是无标签数据集中的自变量数据集、*x*3 指的是预测数据集中的自变量数据集、*x_all* 指的是所有自变量数据集的合并（也就是 *x*1、*x*2、*x*3 在行方向上的堆叠）、*y* 指的是带标签数据集中的因变量。

2 解决过程

2.1 数据集缺失情况以及处理办法

2.1.1 数据集缺失情况

指标缺失率如下，表格中的数值指的是缺失率（单位无）。

指标	x1	x2	x3	x_all	指标	x1	x2	x3	x_all
x1	0.0000	0.0000	0.0000	0.0000	x80	0.0000	0.0005	0.0000	0.0002
x2	0.0000	0.0000	0.0000	0.0000	x81	0.0097	0.0204	0.0004	0.0102
x3	0.5126	0.4455	0.4386	0.4656	x82	0.0031	0.0029	0.0025	0.0028
x4	0.5126	0.4455	0.4386	0.4656	x83	0.0031	0.0029	0.0025	0.0028
x5	0.5126	0.4455	0.4386	0.4656	x84	0.0031	0.0029	0.0025	0.0028
x6	0.5126	0.4455	0.4386	0.4656	x85	0.0031	0.0029	0.0025	0.0028
x7	0.5126	0.4455	0.4386	0.4656	x86	0.0031	0.0029	0.0025	0.0028
x8	0.5126	0.4455	0.4386	0.4656	x87	0.0031	0.0029	0.0025	0.0028
x9	0.4897	0.4521	0.4462	0.4627	x88	0.0031	0.0029	0.0025	0.0028
x10	0.4897	0.4521	0.4462	0.4627	x89	0.0031	0.0029	0.0025	0.0028
x11	0.4897	0.4521	0.4462	0.4627	x90	0.0051	0.0090	0.0016	0.0052
x12	0.4897	0.4521	0.4462	0.4627	x91	0.0451	0.0586	0.0011	0.0349
x13	0.4897	0.4521	0.4462	0.4627	x92	0.9986	0.9652	1.0000	0.9879
x14	0.4897	0.4521	0.4462	0.4627	x93	0.7174	0.7366	0.7102	0.7214
x15	0.5126	0.4455	0.4386	0.4656	x94	0.9986	0.9652	1.0000	0.9879
x16	0.5126	0.4455	0.4386	0.4656	x95	0.1809	0.2446	0.0374	0.1543
x17	0.5126	0.4455	0.4386	0.4656	cat_x96	0.0000	0.0000	0.0000	0.0000
x18	0.5126	0.4455	0.4386	0.4656	cat_x97	0.0099	0.0213	0.0051	0.0121
x19	0.5126	0.4455	0.4386	0.4656	cat_x98	0.0099	0.0213	0.0051	0.0121
x20	0.5126	0.4455	0.4386	0.4656	cat_x99	0.0099	0.0213	0.0051	0.0121
x21	0.4897	0.4521	0.4462	0.4627	cat_x100	0.0099	0.0213	0.0051	0.0121
x22	0.4897	0.4521	0.4462	0.4627	cat_x101	0.0099	0.0213	0.0051	0.0121
x23	0.4897	0.4521	0.4462	0.4627	cat_x102	0.9999	0.9999	0.9998	0.9999
x24	0.4897	0.4521	0.4462	0.4627	cat_x103	0.9997	0.9999	0.9993	0.9996
x25	0.5126	0.4455	0.4386	0.4656	cat_x104	0.9993	0.9987	0.9992	0.9991
x26	0.5126	0.4455	0.4386	0.4656	cat_x105	0.9981	0.9983	0.9978	0.9981
x27	0.5126	0.4455	0.4386	0.4656	cat_x106	0.9993	0.9995	0.9989	0.9992
x28	0.5126	0.4455	0.4386	0.4656	cat_x107	0.9999	0.9883	0.9999	0.9960
x29	0.5126	0.4455	0.4386	0.4656	cat_x108	0.9997	0.9999	0.9997	0.9998
x30	0.5126	0.4455	0.4386	0.4656	cat_x109	0.9997	0.9999	0.9995	0.9997
x31	0.4897	0.4521	0.4462	0.4627	cat_x110	1.0000	0.9995	1.0000	0.9998
x32	0.4897	0.4521	0.4462	0.4627	cat_x111	0.9997	0.9999	0.9997	0.9998
x33	0.4897	0.4521	0.4462	0.4627	cat_x112	1.0000	0.9998	1.0000	0.9999
x34	0.4897	0.4521	0.4462	0.4627	cat_x113	0.9999	0.9976	1.0000	0.9992
x35	0.4897	0.4521	0.4462	0.4627	cat_x114	0.9999	1.0000	0.9999	0.9999
x36	0.4897	0.4521	0.4462	0.4627	cat_x115	0.9998	0.9998	0.9998	0.9998
x37	0.4897	0.4521	0.4462	0.4627	cat_x116	1.0000	0.9922	0.9999	0.9974
x38	0.4897	0.4521	0.4462	0.4627	cat_x117	0.9997	0.9998	0.9995	0.9997
x39	0.0085	0.0215	0.0051	0.0117	cat_x118	0.9999	0.9994	1.0000	0.9998
x40	0.0085	0.0215	0.0051	0.0117	cat_x119	0.9998	0.9996	0.9999	0.9998
x41	0.0085	0.0215	0.0051	0.0117	cat_x120	0.9991	0.9997	0.9990	0.9993
x42	0.0085	0.0215	0.0051	0.0117	cat_x121	0.9985	0.9975	0.9986	0.9982
x43	0.0085	0.0215	0.0051	0.0117	cat_x122	0.9987	0.9990	0.9986	0.9988
x44	0.0085	0.0215	0.0051	0.0117	cat_x123	0.9999	0.9984	0.9999	0.9994
x45	0.0085	0.0215	0.0051	0.0117	cat_x124	0.9993	0.9993	0.9987	0.9991
x46	0.0085	0.0215	0.0051	0.0117	cat_x125	0.9999	0.9887	0.9992	0.9959
x47	0.0085	0.0213	0.0051	0.0116	cat_x126	0.9999	1.0000	1.0000	1.0000
x48	0.0099	0.0213	0.0051	0.0121	cat_x127	0.9997	0.9999	1.0000	0.9999
x49	0.0085	0.0213	0.0051	0.0116	cat_x128	0.9998	0.9999	0.9999	0.9999
x50	0.0085	0.0213	0.0051	0.0116	cat_x129	1.0000	0.9995	0.9999	0.9998
x51	0.0085	0.0213	0.0051	0.0116	cat_x130	0.9998	1.0000	1.0000	0.9999
x52	0.0085	0.0213	0.0051	0.0116	cat_x131	0.9999	0.9996	0.9997	0.9997
x53	0.0085	0.0213	0.0051	0.0116	cat_x132	1.0000	0.9975	0.9999	0.9991
x54	0.0085	0.0213	0.0051	0.0116	cat_x133	0.9999	0.9999	0.9998	0.9999
x55	0.0085	0.0213	0.0051	0.0116	cat_x134	1.0000	0.9928	0.9999	0.9976
x56	0.0085	0.0213	0.0051	0.0116	cat_x135	0.9999	0.9999	1.0000	0.9999
x57	0.0085	0.0213	0.0051	0.0116	cat_x136	0.9995	0.9996	0.9989	0.9993
x58	0.0099	0.0213	0.0051	0.0121	cat_x137	0.9999	0.9997	0.9999	0.9998
x59	0.0085	0.0213	0.0051	0.0116	cat_x138	0.9999	1.0000	1.0000	1.0000
x60	0.0085	0.0213	0.0051	0.0116	cat_x139	0.0003	0.0011	0.0000	0.0005
x61	0.0085	0.0213	0.0051	0.0116	cat_x140	0.6431	0.7839	0.6322	0.6864
x62	0.0085	0.0213	0.0051	0.0116	cat_x141	0.6435	0.7850	0.6323	0.6869
x63	0.0085	0.0213	0.0051	0.0116	cat_x142	0.6651	0.7995	0.6551	0.7066
x64	0.0085	0.0213	0.0051	0.0116	cat_x143	0.0031	0.0029	0.0025	0.0028
x65	0.0085	0.0213	0.0051	0.0116	cat_x144	0.0031	0.0029	0.0025	0.0028
x66	0.0085	0.0213	0.0051	0.0116	cat_x145	0.0031	0.0029	0.0025	0.0028
x67	0.0085	0.0213	0.0051	0.0116	cat_x146	0.0031	0.0029	0.0025	0.0028
x68	0.0085	0.0213	0.0051	0.0116	cat_x147	0.0031	0.0029	0.0025	0.0028
x69	0.0085	0.0213	0.0051	0.0116	cat_x148	0.0031	0.0029	0.0025	0.0028
x70	0.0085	0.0213	0.0051	0.0116	cat_x149	0.0031	0.0029	0.0025	0.0028
x71	0.0085	0.0213	0.0051	0.0116	cat_x150	0.0031	0.0029	0.0025	0.0028
x72	0.0085	0.0213	0.0051	0.0116	cat_x151	0.0031	0.0029	0.0025	0.0028
x73	0.0085	0.0213	0.0051	0.0116	cat_x152	0.0031	0.0029	0.0025	0.0028
x74	0.0085	0.0213	0.0051	0.0116	cat_x153	0.0317	0.1031	0.0001	0.0450
x75	0.0085	0.0213	0.0051	0.0116	cat_x154	0.0051	0.0090	0.0016	0.0052
x76	0.0085	0.0213	0.0051	0.0116	cat_x155	0.0051	0.0090	0.0016	0.0052
x77	0.0085	0.0213	0.0051	0.0116	cat_x156	0.0596	0.0760	0.0012	0.0456
x78	0.0099	0.0213	0.0051	0.0121	cat_x157	0.3957	0.4172	0.0362	0.2830
x79	0.0085	0.0214	0.0051	0.0117					

其中指标缺失率分为3个档次，最严重为99%左右，中等的大概是60%~70%，较低的是45%左右，其中最后一个指标缺失率在平均来说在30%左右。其余指标有略微缺失，可以忽略不计。拿到这份数据，寻常想法是不是删除其中最严重缺失档次的指标？都缺失了百分之九十九还要他干嘛。本人一开始也是这么处理的，但是经过**尝试**，发现如果仅仅通过缺失率来删除掉这些指标，然后相对于*XGBoost*模型在同样设定下来说成绩**似乎有一点点下降**，而删除之后节省的成本又可以忽略不计，所以在此步骤中予以保留。

当然以上一段话都是对于*XGB*、*LightGBM*等集成的决策树模型来说的。本人没有怎么尝试过神经网络、*SVM*等其他类型的模型，因为这些模型在特征维度相对于样本数来说过高的话，效果会不好。

2.1.2 处理办法

不直接依靠缺失值率高低而进行直接删除，不意味着不对指标进行筛选，本人对指标筛选的手段是依靠*XGBoost*计算得到的特征重要性——也就是*XGBoost*在迭代过程中对分类起的作用性大小。这也是**尝试**所得。

2.2 类别型变量

自变量数据集中从*x*96开始到*x*157都是类别型变量，也就是相对之间没有大小关系，就像性别变量一样。对于此处理方法有很多，比如*Onehot*编码。但是本文中对此没有进行任何处理，是什么样就是什么样。这是**尝试**所得，如果进行*Onehot*编码，那么相对于*XGBoost*模型在同样设定下来说成绩**似乎有一点点下降，**本人**记得大概在一两个千分位。**所以本文中对类别型变量予以不处理。

2.3 *XGBoost*

数据和指标：*x*1和*y*；所有指标。

调参：

1、设定初始参数和初始化最大树的数量：

初始化参数。

objective='*binary*:*logistic*',*booster*='*gbtree*',*learning_rate*=0.1,*n_estimators*=1000,*gamma*=0,*max_depth*=5,*min_child_weight*=1,*subsample*=0.8,*colsample_bytree*=0.8,*reg_alpha*=0,*reg_lambda*=1,***scale_pos_weight***=1,*n_jobs*=-1

其中***scale_pos_weight***设定为1，如果设定为其他正值，比如正样本数除以负样本数的值，那么就会使得成绩下降2到3个千分点，相对于同样设定下的*XGBoost*来说。其余除了*max_depth*=5，*subsample*=0.8，*colsample_bytree*=0.8以外都是默认值，至于为何设定为这样，这是直觉，一般是这样子，*max_depth*一般在4-6之间都是不错的初始值。

在设定为初始化参数的情况下，再使用*Xgboost*模块中的方法*cv*（*xgboost.cv*），设定：*params=para_best,dtrain=xgboost.DMatrix*(*x*1,*y*),*num_boost_round*=1000,*nfold*=4,*stratified=True,metrics*='*auc*',*early_stopping_rounds*=20,*as_pandas=True,verbose_eval=False,show_stdv=True*。

其中*para_best*就是前面的初始化参数，运用四折交叉验证（*nfold*=4），四折交叉验证的时候生成每一折的数据都是分层抽样，这是为了控制每一折中的正负样本比例与原始数据接近（*stratified=True*），这样子应该是比较好的。什么时候停止迭代呢？在此轮下以后的20轮，也就是再集成20棵决策树的时候（*early_stopping_rounds*=20），四折交叉验证的验证集的*auc*（*metrics*='*auc*'）的平

均值不再提升的时候就停止，返回这个时候的决策树的数量。

初始化了参数和得到了初始化参数下 *cv* 得到的最大树的数量，这就是一个起始点，开始后面的调参。

2、网格搜索：

一般来说，本人喜欢调试的参数是：*max_depth*、*min_child_weight*、*subsample*、*colsample_bytree*、*reg_alpha*、*reg_lambda*。最大深度和最小叶子节点权重是用于控制决策树的结构和过拟合的，列采样率和行采样率来源于 *Bagging* 思想，*L*1 和 *L*2 正则是来控制过拟合。这六个参数对于结果是影响很大的，至于 *gamma* 也有影响，但是一般来说在默认的 0 处就有很好表现，所以就不放置于调参的参数中了。六个参数一起调整，维度已经很爆炸了，只是这次比赛的数据量不大，指标也不多，所以可以一起调。

通过 *sklearn* 模块中的 *GridSearchCV* 方法，跑了五六个小时以后，得到最优参数。

3、降低学习率

在学习率为 0.1 的情况下，学习的是很快，但是精度当然比不上细水长流来的好，所以这里开始降低学习率（经过尝试为 0.008 的时候比较好，不知道是不是最好），其余参数值不变。也是通过 *xgboost* 模块中的 *cv* 方法确定最优的树的最大数量。然后万事俱备，拿来预测 *x*3。

3、结果：*B* 榜 0.7537，*A* 榜应该是 0.752。

说明，在第 1 步和第 3 步是借用的 *Xgboost* 模块本身 *cv* 方法，而第 2 步因为要使用网格搜索，所以需要调用 *Xgboost* 中的 *sklearn* 接口模块：*XGBClassifier*。最后的预测也是用的 *XGBClassifier*。

2.4 特征筛选

上一步中得到了调优参数下的 *XGBoost* 模型，可以通过其中的 *feature_importances_*方法得到指标的重要性：

指标	重要性	指标	重要性	指标	重要性	指标	重要性
x80	0.1060	x72	0.0098	x16	0.0014	x106	0.0000
x95	0.0451	x47	0.0096	x76	0.0014	x107	0.0000
x2	0.0383	x68	0.0092	x87	0.0014	x108	0.0000
x81	0.0292	x57	0.0090	x96	0.0012	x109	0.0000
x63	0.0274	x43	0.0088	x37	0.0011	x110	0.0000
x157	0.0255	x69	0.0088	x139	0.0010	x111	0.0000
x54	0.0232	x45	0.0087	x75	0.0009	x112	0.0000
x52	0.0230	x78	0.0084	x77	0.0009	x113	0.0000
x62	0.0223	x64	0.0073	x22	0.0008	x114	0.0000
x61	0.0220	x44	0.0070	x11	0.0006	x115	0.0000
x93	0.0220	x46	0.0070	x143	0.0004	x116	0.0000
x1	0.0210	x58	0.0069	x144	0.0004	x117	0.0000
x40	0.0196	x99	0.0067	x85	0.0003	x118	0.0000
x30	0.0192	x91	0.0065	x86	0.0003	x119	0.0000
x14	0.0190	x153	0.0057	x5	0.0002	x120	0.0000
x8	0.0175	x67	0.0056	x17	0.0002	x121	0.0000
x19	0.0173	x154	0.0054	x150	0.0002	x122	0.0000
x24	0.0172	x60	0.0053	x15	0.0001	x123	0.0000
x48	0.0151	x82	0.0052	x31	0.0001	x124	0.0000
x20	0.0149	x83	0.0052	x3	0.0000	x125	0.0000
x35	0.0146	x26	0.0049	x6	0.0000	x126	0.0000
x7	0.0145	x142	0.0048	x9	0.0000	x127	0.0000
x97	0.0145	x84	0.0046	x12	0.0000	x128	0.0000
x36	0.0142	x32	0.0043	x18	0.0000	x129	0.0000
x42	0.0138	x73	0.0043	x21	0.0000	x130	0.0000
x140	0.0138	x65	0.0034	x23	0.0000	x131	0.0000
x41	0.0134	x66	0.0032	x25	0.0000	x132	0.0000
x59	0.0129	x4	0.0026	x28	0.0000	x133	0.0000
x29	0.0128	x39	0.0025	x34	0.0000	x134	0.0000
x51	0.0128	x74	0.0021	x70	0.0000	x135	0.0000
x156	0.0127	x98	0.0021	x71	0.0000	x136	0.0000
x53	0.0121	x155	0.0021	x89	0.0000	x137	0.0000
x79	0.0121	x27	0.0020	x92	0.0000	x138	0.0000
x13	0.0119	x49	0.0019	x94	0.0000	x145	0.0000
x50	0.0117	x88	0.0019	x102	0.0000	x146	0.0000
x141	0.0117	x100	0.0019	x103	0.0000	x147	0.0000
x56	0.0108	x101	0.0018	x104	0.0000	x148	0.0000
x55	0.0105	x33	0.0016	x105	0.0000	x149	0.0000
x38	0.0104	x90	0.0016			x151	0.0000
		x10	0.0014			x152	0.0000

可以看出指标的重要性呈现指数型递减特征，似乎所有的集成的树模型都是如此，从 *x*75 开始，之后的指标都是可有可无的，毫无用处，所以为了精简结构和提升效率，本人设定在权重在 0.0001 的指标统统删除。至于为何设定阈值是 0.0001，是不是看起来 *x*85 到 *x*31 的特征权重不过在 0.0001、0.0002 左右，也很小很小，为何不将其删除。因为它们至少还有一点点用处，没必要删掉，似乎经过尝试删掉要使得成绩下降一点点。而且树模型对指标维度相对于样本数的比重要求不高（树模型的弊病是很容易过拟合，*XGB* 在此基础上加入正则项等一系列超参数就是此缘由），所以就结合成绩下降一点点的表现，就予以保留，只是把完全无用的删除掉。**其中删除的指标群中，缺失率极高的指标几乎都囊括在内。所以说这就是之所以不急着在前面删除，而在这利用 *XGBoost* 挑选。而且经过尝试这样子效果也要好些。模型筛选一般来说好于人为筛选。**

经过特征筛选，还剩下 98 个指标。

筛选之后再次运行 *XGBoost*，成绩 *B* 榜达到 0.75501，*A* 榜似乎也到了 0.753。听说于单模型 *LightGBM* 跑到 *B* 榜 0.76 几的那位大神来说，本人这单模型 *XGBoost* 的成绩还算过得去吧。

2.5 再次调参

前面经过特征筛选，数据集已经发生变化，而再用原始数据集调优参数，似乎不太好。

实际上，如果再用原始数据集调优参数，不进行新一轮地调参，接着进行后面的迭代半监督，然后再预测，上传结果，就已经到了 *B* 榜第一了，*AUC* 是 0.7582。（说明一下：现在的 *B* 榜第二队伍

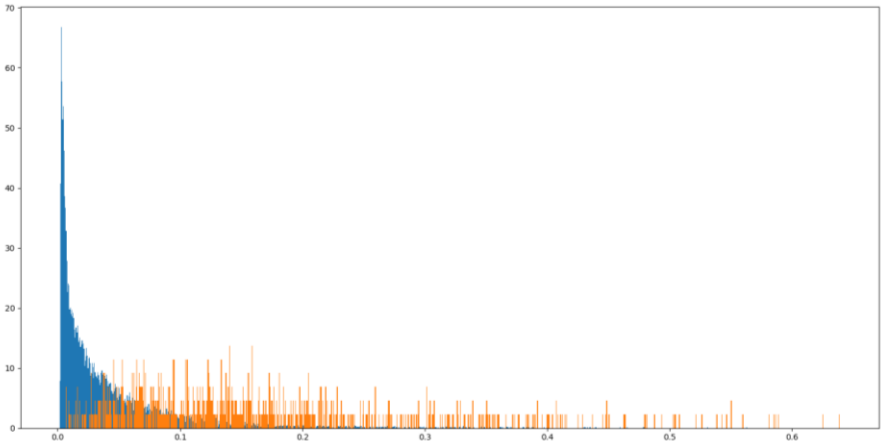
*weChatoqf*9*oxK*8*vLHJeD*1*blAKVcFVKFGfk*2*dbab*，他们从 70 多名冲上来是因为本人在他们账号里上传了在不再次调参情况下的预测结果。因为本人要上传的时候 *DC* 网站抽风了，所以借用他们的账号上传下，他们也是西财的，都是本人的好朋友）。

所以进行新一轮参数调优，这里的调参思路与前一轮一致，不再赘述。

2.6 迭代半监督

到重头戏了，前面进行新一轮调参，*AUC* 上升了一些，到达了 0.7553，变动幅度不大。这也是正常的，本人觉得，如果一个模型，没有指标上和训练数据样本数上的变化，是很难有所大的提升的。所谓的特征筛选和构造、半监督学习就是如此，为了创造更加有解释力的变量，或者将没有解释力的变量剔除掉，和加入更多有学习潜力的样本，这样子才能让模型上升的幅度较大，至少是 *AUC* 值千分位的上升。当然有人说，调参也可以调整到上升若干个千分位——在现有数据集和指标下，是模型本身的上限就已经包含了那个水平，你只是在逼近而已。而特征和样本是让模型的上限提升。所谓特征和样本决定上限，调参决定下限应该就是如此吧。

这里迭代半监督的方法是：本人前面的 *XGBoost* 已经达到了 0.7553，个人觉得，已经是个很好的水平了。这里可以使用模型预测 *x*1、*x*2 和 *x*3，得到所有样本对应的高风险用户的概率水平（如果这里看不懂，可以好好学习下 *Logistics* 回归和 *XGBoost* 模型）。其中 *x*1 对应的样本的高风险用户的概率水平是有对应的标签的，其不同类型（0 或者 1）下的预测概率水平如下图所示：



其中蓝色的是低风险用户（*y* 为 0）的直方图，黄色的是高风险用户（*y* 为 1）的直方图。本人不清楚新网银行确定高低的阈值是多少，所以就只有像网格搜索一样去穷举出所有可能的点，将 *x*2 和 *x*3 的样本的预测概率水平在改点以上用户划定为高风险用户，然后带入到模型进行训练。至于为何不带划定为低风险的用户进行训练，因为低风险的用户的样本数量已经足够多了，而且经过尝试，本人曾经以双阈值进行划分，取线下最优的划定方式进行划定样本带入训练，结果不升反降。所以借鉴过抽样的思想，就只将划定为 *x*2 和 *x*3 中高风险用户样本带入进行训练。这个阈值经过线下搜索和线上尝试，一系列证据表示阈值在 0.3 时，多一点，少一点线上和线下的 *AUC* 都会下降。本人有预感，新网银行应该也是以该阈值划分高低风险用户类型的。

带入从 *x*2 和 *x*3 数据集中更具预测概率划定的高风险用户的样本，和 *x*1 数据集全部放入经过特征筛选和两轮调参的 *XGBoost* 中。结果使得 *AUC* 值上升到：0.75973，再次提升，暂居第一。

3 心得体会

这一系列步骤走来，看起来很写意轻松，每一步都是理所应当，好像都顺其自然，简简单单。但是，其中不断地尝试，对原始数据集与与处理方法地处理尝试，参数不断调优（在两轮调参中，本人是网格搜索和人为设定结合调试的），方法上的不断尝试下走出的一条路，整个比赛也是本人独立完成，其中有多少辛酸，都不一而足，相信每一位认真参加比赛的同仁应该可以体会，夜深人静之时、毫无进展之时，不觉仰望星空，觉得要是自己可以直接卜算出最优参数该多好，省的调调调……最后发现，好像以前想的那些复杂的，高端的都其实没有用——殊途同归，大道至简。

至于模型融合的问题，本人个人觉得，模型融合的宗旨是：好而不同。不但要有差异性，还要都是好的。如果只有差异性，那真的太好达到了。周志华的《机器学习》中曾经经过简单证明，如果 *T* 个模型，每个都是独立同分布的，注意是独！立！同！分！布！要求是独立的。那么这 *T* 个模型如果预测正确率都在 0.5 以上，那么就会使得正确率在 *T* 趋近于正无穷的情况下最后结果依概率收敛到 1。这个证明告诉本人们，特么不要相信什么 0.5 正确率以上的独立模型还依概率收敛到 1，开玩笑吧，还正无穷个模型，你咋不上天呢。额，只是开个玩笑而已哈，没有其他意思，纯粹当发个牢骚。就因为这个定律，本人以及相信很多的参赛者都在辛辛苦苦地调各个模型，有的还为了差异性将训练的数据集进行各种各样千奇百怪地变化。最终本人顿悟了，本来变化就是不对的，已经明确尝试了出来不填缺失值比填了缺失值要好，还拿着填了缺失值的去融合，这不跟自己找不自在吗？样本数量这么少，相对于指标数量来说，还去拿神经网络、*SVM* 来做融合，这不是喝多了吗？当然，类似的不少事情，本人也干过……

当然，本人不是表明模型融合没有用。真的有用！本人敢断定，如果还添加一些做的好的 *LightGBM* 等模型来和这里的 *XGBoost* 融合，相信最后的 *AUC* 还能提升，只是在本人这里新网杯已经翻篇了。而且 *LightGBM* 也很难调参，比 *XGBoost* 还要难调。

比赛就是如此，不断地尝试和想法的实现中使得结果走得越来越好，当然本人这个成绩绝对不是最好的，希望有识之士一起交流。至于本人为何没有出现在复赛，是因为本人是第一次参加比赛，没有看清比赛规则，忘记提交了 *B* 榜文件，测出来 *A* 榜中最好的到了当时 *B* 榜第九了。哎，多么痛的领悟。希望读者一定要引起教训。

4 再次提升方向推荐

1、暴力半监督

可以参考“吾王不可直视”的 *GitHub* 文档（*https://github.com/wepe/DataCastle-Solution*）。本人觉得他整篇文档唯一可取之处就是迭代半监督，其他什么“计数特征”、“离散特征”、“缺失值特征”、“排序特征”都是扯淡的。平白无故扩大了样本维度，增加了训练和调参难度。将本人带到沟里去了，还好本人反省了，返本朔源，重新开始了。

暴力半监督在这里只是纯粹为了提升成绩而来的，在现实生活中根本操作不了，家里有矿都不管用。想要加入 10 个样本就要拿模型训练 1024 次，这怎么搞得下去？不过这次样本数量和特征都不多，所以有一定的可操作性。

2、模型融合（*LightGBM*、*CatBoosting* 等）

可以试着跟 *XGBoost* 融合下提高预测水平，前提是“好而不同”，不要拖 *XGBoost* 模型的后腿，融合方式可以是加权融合或者是 *Stacking*。

3、交互特征

可以通过模型 *XGBoost* 得到特征重要性，然后取前若干名特征，进行交互：*xy*，*x+y* 等方式，形成新的特征带入训练，注意构造特征要三份数据集一起动。