参赛队队名：wahahaha

竞赛报告书

一、 参赛作品概述

1） 赛题分析

赛制提供的数据为：

带标签数据集 15000 个样本，测试集 5000 个样本。数据集共有 1138 个特征，有 numeric 类型和 category 类型，评价指标为 AUC。

数据集输出类型为 0（人品堪忧）和 1（人品优秀），是个二分类问题。测试集输出为概率，在（0,1）之间。

2） 数据分析

从数据集查看，有一些 numeric 特征存在大量缺失值，以及个别样本缺失值的缺失情况严重，可以考虑将这部分数据删除，以及对特征进行排序处理。另外，category 类型可以采用 One-Hot Encoder 处理。

3） 训练模型

主要使用到 xgboost 和 svm 模型。xgboost 单模型采用 Boosting 机制，可以训练出很好的表现形式，而 svm 采用的是 Bagging 机制，可作为模型差异补充。除了调参工作，不同语言（Python，R 语言）的 Xgboost 训练得出来的分数不一样，并且存在一定的差异。

4） 模型融合

利用模型间的差异性，对多个模型的结果进行融合，最终得到的结果要比单模型训练的出来的效果要好很多。我们采用的方法是基于正序 rank 进行加权融合。

二、参赛作品技术路线

1. 算法总思路
   A. 数据预处理：
      1） 对 category 类型的数据采用 One-hoe-encoder 处理（独热编码，特征 x411、x415~x417、x1107~x1138）。
      2） 部分样本特征存在大量的缺失值，把将缺失值达到 190 的样本进行删除，部分 numeric 特征存在大量的缺失值，把缺失值达到 10000 的样本进行删除。
      3） 对特征进行从小到大排序，得到的排名为排序特征。采用 Xgboost 对特征进行排名，排名在前 600 的排序特征和原始特征抽出来进行组合训练。
      4） 这里考虑到剩下的缺失值处理问题，经过多次训练发现，对剩下的缺失值不做处理的效果要好于取均值和众数，因此最后选择对剩余的缺失值不做处理。
   B. 模型训练：运用了 R 和 Python 两种语言使用不同参数的 xgboost 对上述生成的不同数据集进行处理，并通过各种参数的调整找到对应的该模型的最高分的参数集合。以及同样得到方法使用 SVM 对模型进行单独训练。
   C. 模型融合与进一步融合：由于单模型训练的程度有限，我们试图通过模型融合来找到突破，采用 MIC 机制对多个数据结果进行分析，找到三个差异性最大

的模型，并使用不同的参数进行加权融合。并使用上述最终的最好结果与Xgboost（Python）所得的最好结果进行进一步融合，调整权重后得出最终结果。

2. 算法原理

1） 特征预处理

a) 统计训练集特征缺失值，将缺失值达到 10000 以上的特征进行删除，；
b) 统计训练集样本缺失值，将缺失值达到 190 的样本进行删除；
c) one-hot-encoder，将 category 类型特征进行独热编码；
d) 特征排序，对 numeric 特征进行从小到大排序；
e) 特征选择，采用 xgboost 对特征进行排名，排名在前一定名次的单独抽出来训练。

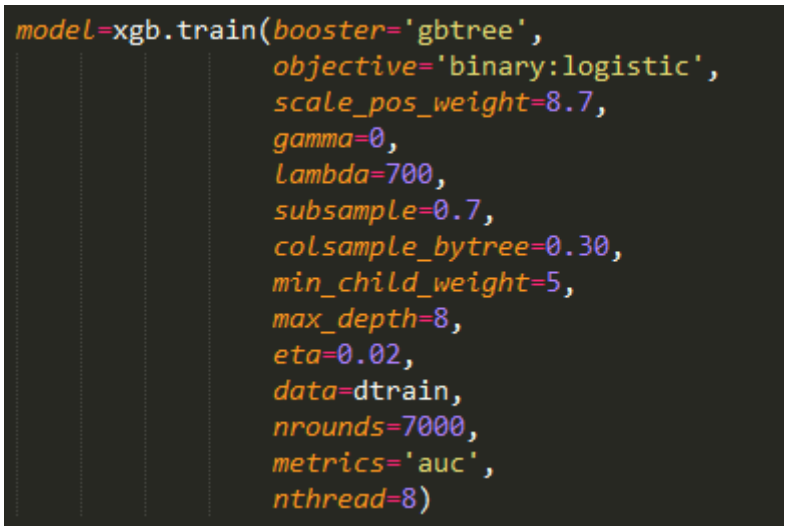
2） Xgboost（Python）

a) 对数据不做处理，训练后线上得分为 0.7014
b) 对数据特征缺失值和样本缺失值较多的分别进行特征和样本删除，线上得分为 0.7094
c) 对 category 特征进行独热编码，训练后线上得分为 0.7111
d) 不同的数据处理之后训练所得的结果中，最高得分为 0.7111。于是试图在不同的语言上寻求突破。

3） Xgboost（R）

a) 对数据不做处理，训练后线上得分为 0.7144
b) 对数据特征缺失值和样本缺失值较多的分别进行特征和样本删除，线上得分为 0.7161
c) 对 category 特征进行独热编码，训练后线上得分为 0.7211
d) 调整训练次数为 1300，线上得分 0.7144，调整训练次数为 5200，分数为 0.718，调整训练次数为 7000，分数为 0.7211，再分别调整训练次数为 7500/8000/8700，得到最终最高的分数为 0.7211。

参数设置如下：

Xgboost（Python，线上得分 0.7111）	
Xgboost（R 语言，线上得分 0.7211）	

```
params={
    'booster':'gbtree',
    'objective': 'binary:logistic',
    'early_stopping_rounds':100,
    'scale_pos_weight': 6.18,
        'eval_metric': 'auc',
    'gamma':0.1,
    'max_depth':8,
    'lambda':550,
        'subsample':0.7,
        'colsample_bytree':0.4,
        'min_child_weight':3,
        'eta': 0.02,
    'seed':random_seed,
    'nthread':8
    }
```

```
model=xgb.train(booster='gbtree',
                objective='binary:logistic',
                scale_pos_weight=8.7,
                gamma=0,
                lambda=700,
                subsample=0.7,
                colsample_bytree=0.30,
                min_child_weight=5,
                max_depth=8,
                eta=0.02,
                data=dtrain,
                nrounds=7000,
                metrics='auc',
                nthread=8)
```

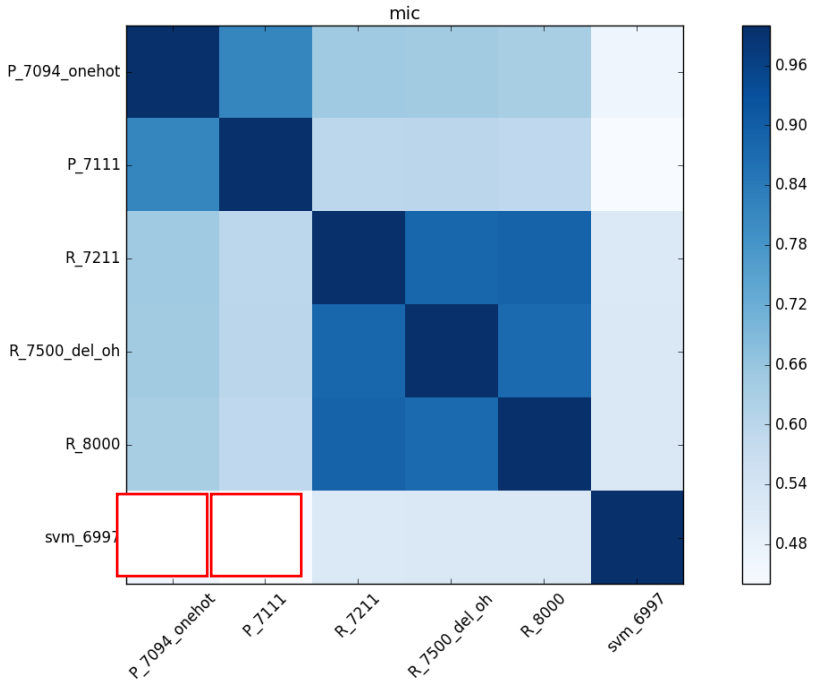
4） SVM

xgboost 单模型采用 Boosting 机制，可以训练出很好的表现形式，而 svm 采用的是 Bagging 机制，可作为模型差异补充。

a) 对数据不做处理。训练后线上得分为 0.676（中间出现过拟合现象）
b) 对数据特征缺失值和样本缺失值较多的分别进行特征和样本删除，线上得分为 0.6809
c) 对数据特征进行排序，训练后线上得分为 0.6997
d) 不同的数据处理之后训练所得的结果中，最高得分为 0.6997。

5） 模型融合

由于单模型训练的程度有限，我们试图通过模型融合来找到突破。为了直观地展现出模型之间的差异性，采用 MIC 机制对多个数据结果进行分析。（其中颜色越浅，表示差异性越大）



由上图可得，红色框的部分表示模型之间差异性较大，因此我们采用这三个模型进行加权融合。下面是实验中的部分加权融合的方案：

```
score = 0.5*R7211.score + 0.28*P7111.score + 0.22*svm6997.score 7223
score = 0.6*R7211.score + 0.2*P7111.score + 0.2*svm6997.score 7222
score = 0.7*R7211.score + 0.2*P7111.score + 0.1*svm6997.score 7221
```

经过多次的权重系数调整，得到该融合模型的最佳方案，得分为 0.7223。

6） 进一步融合

将上述所得的最佳结果 Xgboost_0.7223 与 Python 语言所得的最佳结果 Python_0.7111 进一步融合，调整权重系数后，0.618*Xgb_7223+0.382*Python_7111，得到最终结果 0.723。

三、作品总结

1. 优势总结

1） 可以简便的与新的模型进行加权融合，扩展性较强。

2） 主要是对数据进行预处理，提高数据的可靠性；

3） 使用不同语言的 xgboost 版本，可以得到不同的训练分数；

4） 对训练结果进行加权融合，要比单模型的训练分数高；

2. 可能改进的方向

1） 我们没有对无标签样本进行采集，可以在选择一些无标签样本加入到训练集进行模型提升；

2） 随机深林、神经网络算法可以作为模型参考；

3） 删除缺失值特征和样本后，我们的训练分数没有得到大幅度提升，由于时间原因，可以进一步对其优化。

4） 训练时间较长，没有做过多的参数调整。

5） 参考其他人的分享，对 numeric 特征进行等量离散化处理，但是分数反而下降，

可能是相关方法或者参数需要调整。