建模大赛全量模型说明

队名：海尔炮

队员： 崔　灏 中央财经大学　　研究生

李宇杰 北京航空航天大学　研究生

1 目录

2 概述

2.1 比赛目的

本次建模的目的，是根据基于两个相似金融场景的训练数据集，其中一个场景无标签，设计方案对上述两个场景分别开发信用评估模型，在相应场景下，分别预测贷款申请人的违约概率。

根据此次比赛的特点，我们的方案设计如下：

对于 A 场景，基于不同的特征工程方案，得到不同的数据集。在数据集上分别选择最优算法学习，在得到不同模型后进行集成，来充分且有效的利用数据信息。

对于 B 场景，我们针对 B 场景完全无标签的情况，提出了一种全新的稳定的迁移算法.

3 创新点

在本次建模比赛中，我们队伍在多个方面采用了创新的思路，取得了不错的效果。我们将本队伍的创新之处归纳如下：

1、比赛数据的自动化处理。

我们的特征工程实现了对原始数据的自动识别到自动寻优处理、生成特征，没有过多人为的去手动分析每个特征的处理方式。既节约了人工成本，也顺应着自动化机器学习的发展方向。

2、迁移算法（Tradaboost）的原创实现与在 B 数据集上的应用。

在本次比赛中，我们原创实现了一个 Tradaboost 算法，用在 B 集数据上有着很好的效果。

3、多种特征处理方案的交叉融合。

我们对原始数据从五个角度得到了五种不同的处理方案，各自寻找所适合的最佳算法。最后我们创新性地运用了 Stacking，实现了不同数据集的模型 Stacking

4、标准化与可扩展性。

我们的特征工程和算法的设计都重在框架的设计和自动化处理的流程，而弱化了调参、手动识别并指定特征处理方法的作用，因此，本模型非常适用于用在其他的数据集中，只需要很小的改动即可使用。

4 特征工程

4.1 问题的分析

通过对数据的观察，本次比赛所给出的训练集数据有以下几点值得注意的特征：

1、字段数量极多，且都为匿名字段，无法得知其真实含义。

2、 缺失量多，通过计算每个字段的缺失率，发现绝大多数字段的缺失率在99%以上。

3、A 场景有两个多对一表，B 集有一个。需要进行多对一聚合处理

4、字段类型包括类别字段、数值字段、时间日期字段。其中，部分字段是以字符串形式给出的（形如：MC1）。

根据上述观察到的特点，我们本次的特征工程有以下结论：

1、使用自动化处理机制。由于字段数量多（2000 多个），不能逐个手工进行处理，应该在代码中，自动判别字段的类型，并采取相应的处理方式。

2、对同一字段采用多种处理方式，从中判断最优方式。由于无从得知字段的真实含义，而且字段的缺失值填补、多对一聚合等都根据字段含义不同有着多种处理方式（如类别型字段适合用众数填充，而数值型字段适合用中位数或 0 填充），我们的方案是对同一字段采取所有可能的处理办法，再从中选择最优的一种。

3、多种方案处理得到的数据集进行交叉融合。我们得到了多种对数据集处理的方案，由于每种方案都有其缺点也有其独到之处，我们将基于多种方案得到的模型进行融合，以提升模型的性能。

4.2 特征工程流程设计

为了适应自动化批处理的需要，我们将特征工程划分为三个相互独立的流程。

（1）数据清洗（cleaning）

这一流程的目的，是依靠无监督的方法，将从读取到的原始数据，处理成能够被进一步识别并处理的数据类型。这一步会将“字段”（原始数据的每一列）转变为“变量”（可以被 Numpy、Pandas 等标准库识别。）

（2）多对一聚合（aggregating）

这一流程的目的，是从一个 ID 对应多行数据的“变量”，聚合得出每个 ID 对应一行数据的“特征”。

（3）特征提取（transforming）

在经过上面两步之后，我们已经得到了最初步的特征。在这一步，我们会根据所用模型前提条件的不同，对特征进行进一步处理，得到一组新的特征，以求模型能从中提取出最多的信息。

这一流程设计的优点在于，可以在这一构架中同时支持多种特征处理方案，并且最小化了重复计算的工作量。并且，这种标准化的设计，也便于将此处的特征工程稍作改动，迁移至其他数据集中。可以进一步据此，实现一个自动化数据分析的标准模块。

4.3 数据清洗

在这一步中，我们对两种数据进行清洗，包括日期型字段和字符型字段。

4.3.1 日期型字段的处理：

对于几个特殊的日期型字段，我们进行了针对性的处理：

- var16、var17 ： 这两个字段中，既有形如“20160104”的值，也有形如“2016”的值。我们统一将其处理成代表相应年份的四位数。将 0 值记录为缺失值。
- var_06、V_7、V_11 ：这三个字段存在于多对一表中，我们将其转换为 pandas.datetime 数据格式。把形如‘0000-00-00 00:00:00’的数据记录为缺失值

4.3.2 字符型字段的处理：

对于数据类型为字符串的字段，我们发现，在每个字段中，形如“MC20”的字符串里的字母都是一样的，所以，我们只把字符串中的数字提取出来。

4.3.3 缺失字段的剔除

由于有些字段的有效数据很少，这样的字段如果进入模型，会对模型产生很大噪音，还会增加模型的过拟合（Overfitting）风险。所以，我们剔除掉了有效数据低于 10 个的字段。

4.4 多对一聚合

这一步骤针对 ccx 表和 consumer 表进行。在这两个数据表中，变量类型包括了：类别型（即数字不代表实际大小，而是表示所属类别）、数值型（数值有着实际意义）、日期型。每种变量都有不同的处理方式：

4.4.1 类别型变量的处理方式

- 每个类别出现次数占该 ID 总次数的比值
- 出现最多的类别

4.4.2 数值型变量的处理方式：

- 取最大值和最小值。
- 加总
- 求平均值
- 计数

4.4.3 日期型变量的处理方式：

- 统计在一天中各时间段发生的比率。（根据生活经验，我们将时间段划分为了 23 到 9 点、9 点到 17 点、17 点到 23 点三段。）
- 统计在一周中工作日和周末发生的比率。
- 计算了二个相邻时间变量的时间差。
- 最早的行为发生的日期。
- 最晚的行为发生的日期

- 最早日期与最晚日期之间的时间差

4.5 特征提取

4.5.1 方案一：单模型回归特征选择（代号：FS）

在这一方案中，我们会用以下方法遍历每一个特征：

1、用多种不同的方法分别对字段进行处理。方法包括：对类别型特征进行独热（OneHot）编码、对数值型特征进行分箱处理并 OneHot 编码、用多种方法填补缺失值。

2、处理方法的寻优：用每种处理方法得到的特征，对训练集标签进行作逻辑回归（LogisticRegression）拟合，并计算 5 折交叉验证的 auc 分数。在所有处理方法中，AUC 分数最高的哪一种，就是最适合该特征的处理方法。而相应的最高 AUC 的值，就是对该特征的重评分。

3、特征选择：以 AUC=0.525 作为阈值，取所有 AUC 高于这个值的特征。

4.5.2 方案二：不作处理（代号：FN）

如 XGBoost、LightGBM 等基于决策树的完善模型已经内置了对缺失值的处理方法，会自动在不同节点遇到缺失值时采用不同的处理方法，并且会学习未来遇到缺失值时的处理方法。因此，我们在本方案中，不对缺失值作出处理，将经过清洗后的特征直接输入模型。

4.5.3 方案三：基于 XGBoost 特征重要性的选择（代号：FA）

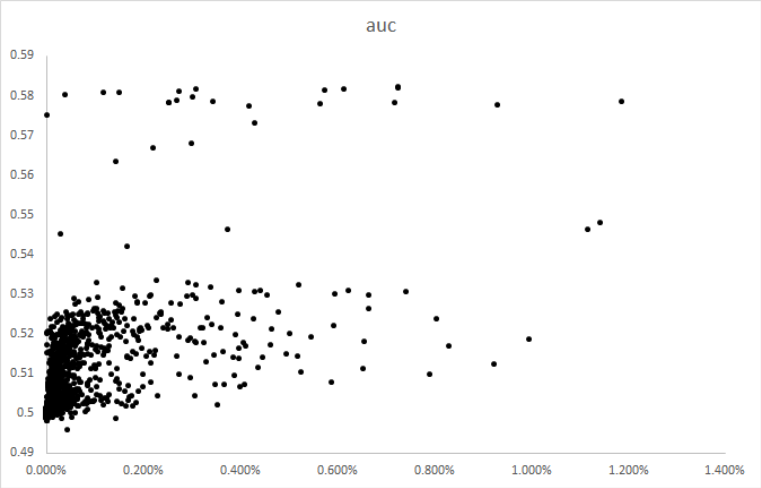
Xgboost 支持在拟合之后导出特征重要性。特征重要性的基本原理是，根据结构分数的增益情况计算出来某个特征的重要性，也就是它在所有树中出现的次数之和。

在这一方案中，我们用随机划分的样本，去训练默认参数的 XGBoost，得到一个特征重要性数据。为了避免随机性和噪声带来的影响，我们用不同的随机种子进行了 500 次运算，最终结果加总求均值。得到对每一特征的排名。

由于清洗后的样本共有 1999 列。根据相似论文中的推荐公式，计算得到最优的特征选取数应该为

$$\sqrt{m} * \log m = 160$$

在本方案中对特征重要性的打分，和方案一中单变量逐步回归得到的 AUC 分数，刻画的散点图如下：



可以看到，因为 LR 和 XGBoost 的假设不同（LR 假设所有变量都是相互独立的，而 XGBoost 会考虑变量之间的关系），所以两种模型对特征的评价有明显差别。

4.5.4 方案四 基于 F-classif 分数的处理方式寻优（代号：FG）

在这一方案中，类似于方案一，我们用 F-classif 分数，来对变量的不同处理方法（用中位数、众数、0 来填补缺失值、OneHot 编码等）作出比较，得到最好的一种处理方式。不对特征作筛选。

4.5.5 方案五 有监督的 IV（信息量）分箱法（代号：FW）

IV 的全称是 Information Value,中文意思是信息价值,或者信息量。其计算公式如下

$$\mathrm{IV_i} = \left({}^{y_i}/_{y_t} - {}^{n_i}/_{n_t}\right)\ln\left(\frac{{}^{y_i}/_{y_t}}{{}^{n_i}/_{n_t}}\right)$$

在评估一组特征（由一列变量分箱得到）的信息量时，其公式为：

$$\mathrm{IV} = \sum IV_i$$

分箱法，指的是将数值型特征的值域划分为若干个数据区间，将每个样本归入相应的区间内。这里采用的是等频分箱法，即每个数据区间内的样本量倾向于一致，以减少样本异常值的影响。

在本方案中，会尝试对每个数值型变量按不同的区间数进行分箱，并 OneHot 编码，选择能使组合的 IV（信息量）最大化的一种分箱方法。最终，再按照 IV 对所有特征进行排序，选取 IV>0.01 的特征。

4.5.6 五种方案的统计

对上述五种方案的统计如下：

方案代号	FS	FN	FG	FA	FW
特征数量 A	263	1999	2424	160	1258
特征数量 B	213	1974	2399	160	1146
筛选方式	单模型拟合	不筛选	不筛选	特征重要性	IV 筛选
寻优方式	单模型拟合	不寻优	F 检验量	不处理	IV 最大化

5 A 集算法说明

5.1 核心算法的介绍

XGBoost 全称是 eXtreme Gradient Boosting。该算法的作者为华盛顿大学的陈天奇（Tianqi Chen）。这是一类将多个弱分类器合成强分类器的 boost 算法，最大的特点是优化目标时使用二阶泰勒展开，同时使用一阶和二阶导数，并且在代价函数中引入了正则项，在过拟合方面有着更好的效果。此外 XGBoost 借鉴了随机森林的列抽样的做法，也能有效防止过拟合并加速计算速度。

LightGBM　全程是 Light Gradient Boosting Machine。该算法由微软 DMTK (分布式机器学习工具包)团队在 GitHub 上开源。该算法相比 XGBoost，在算法、性能上都进行了优化，更适合处理大规模数据集。

Stacking　这是一种集成学习（Ensemble learning）技术。通过袁模型来聚合多个集模型。基础层次模型基于完整的训练集进行训练，然后元模型基于基础层次模型的输出进行训练。能够实现比单个分类器更高的准确率，并且可以提高模型的稳定性，防止过拟合。

5.2 创新的 STACKING：使用不同数据集的模型 STACKING

在此，我们对传统的 Stacking 方式进行了改进，使其支持基于不同数据处理方案训练出来的模型。

通过特征工程，我们得到了五种方案处理的五种不同特征集。为了检验基于不同数据集的模型预测结果的差异，我们使用 XGBoost 对五种数据集的预测结果计算了相关系数：

	FS	FN	FG	FA	FW
FS	1.00	0.93	0.93	0.93	0.92
FN	0.93	1.00	0.94	0.97	0.90
FG	0.93	0.94	1.00	0.94	0.89
FA	0.93	0.97	0.94	1.00	0.91
FW	0.92	0.90	0.89	0.91	1.00

从图中我们可以看出，不同方案数据集的预测结果之间存在着较大的差异性。其中，FW 方案和 FG 方案的预测结果相差最大。

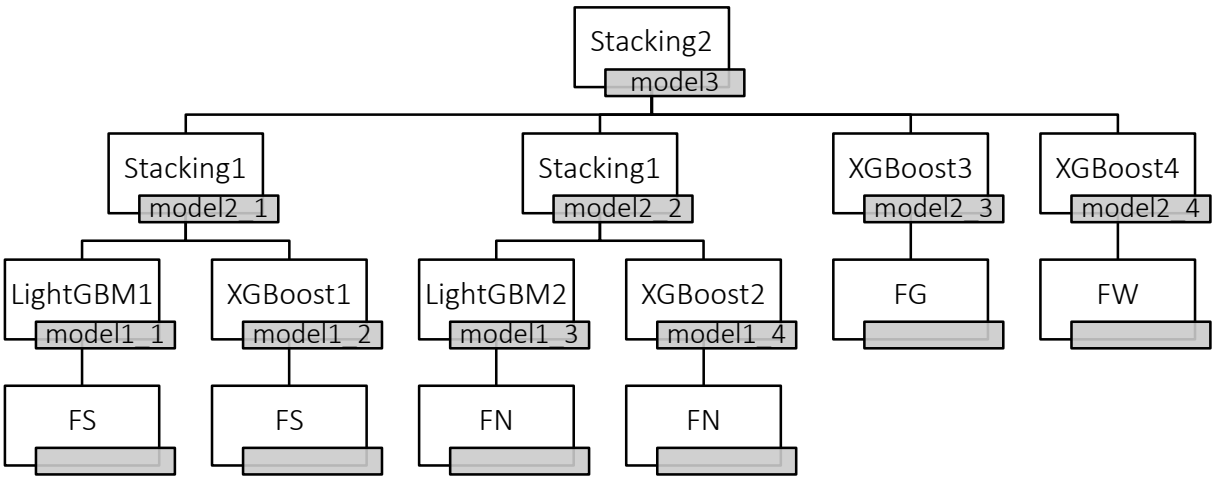
我们又基于不同方案的数据集，用默认参数的 XGBoost 模型，用 2 折交叉验证计算了 AUC 分数，统计如下：

处理方案	XGB 预测分数
FS	0.6453
FN	0.6442
FG	0.6459
FA	0.6507
FW	0.6443

可以看到，虽然不同方案的预测结果差异较大，但是其预测精确度类似。这说明，每种方案都从不同的角度挖掘了数据中的不同信息，并用于模型预测。在这些方案的预测结果上面应用 Stacking，可以综合这些不同角度的信息，从而取得更好的预测能力。

5.3 模型的具体细节

本模型的具体结构如下：



通过对不同算法的尝试，我们最终发现：

对于 FS 和 FN 数据集，使用 XGB 和 LightGBM 两个模型 Stacking 的预测能力较好。

对于 FG、FA、FW 数据集，使用 XGB 单模型有较好的效果。

由于时间所限，我们未在此进行大规模的调参工作。我们仅根据每种方案中特征的数量，相应调整了 XGB 和 LGBM 中的 L1 和 L2 正则化系数，通过限制进入模型的特征数，来防止过拟合。

最后，运用改进过后的 Stacking 方法，我们把基于上述五种特征处理方案的模型进行了进一步的 Stacking，得到了适用于 A 集的最终模型。

6 B 集算法介绍

6.1 算法介绍

迁移学习(Transfer learning) 顾名思义就是就是把已学训练好的模型参数迁移到新的模型来帮助新模型训练。考虑到大部分数据或任务是存在相关性的，所以通过迁移学习我们可以将已经学到的模型参数（也可理解为模型学到的知识）通过某种方式来分享给新模型从而加快并优化模型的学习效率不用像大多数网络那样从零学习

迁移学习的严格定义:

给定源域 Ds={Xs, Fs(X)} 和学习任务 Ts ,目标域 DT={Xt,Ft(X)} 和学习任务 Tt , 迁移学习旨在源域不同于目标域或学习任务 Tt 不同于学习任务 Ts 的条件下通过使用学习任务 Ts 和源域 Ds={Xs,Fs(X)} 所获取的知识来帮助学习目标的在目标域 Dt 的预测函数 Ft(.)

6.2 模型思路

1) 不同于一般的迁移学习，半监督学习，B 场景训练集实际上并不带有信息，没有任何标签。因为得不到 A 场景赋予信息以外的任何信息,B 模型对 B 的预测结果只能趋近于 A 模型对 A 场景预测结果。

2) 把最确定的 B 场景信息加入 A 场景增大 A 场景对 B 场景的预测能力

6.3 迁移算法内容

其中输入数据的意义分别为:**xa**(xa_0，xa_1，x_{a2},…，xa_n)为 A 集训练属性集,**ya**(ya_0，ya_1，ya_2,…，ya_n)为 A 集训练集标签，**xb**(xb_0，xb_1，xb_2,…，xb_n)是 B 场景训练集属性集,范围*(rp,rn)*是一个概率范围,预测概率模型 *model_p* 是由输出概率最准确的算法 *algorithm1* 对（**xa**,**ya**）学习得来的，它对 **xb** 进行预测的预测概率 **p**(p_0，p_1，p_2,…，p_n)以及预测标签集 **yb**（*t-hreshold* 是正负类判断阈

值，每一行数据p_i大于这个阈值即正类，小于这个阈值即负类, *rp>t-hreshold*）. 在其中的数据我们认为是不准确的干扰信息,将其去除，如果在此范围内我们就选择相信它的标签，并将其作为 B 集已知信息**(xb_new,yb_new),**将其通过 *oversampler* 过抽样器过抽样后得到**(xb_new_s,yb_new_s),**我们将它加入 A 集.如下算法，我们将其迭代 N 次，或者知道预测 B 场景中无我们认为的准确信息即$p_{i\in}(rp,rn)$,后停止.此时得到的 A 集用在我们在 A 集交叉验证时最优的模型算法 *algorithm2* 上.得到我们最终模型.

INPUT: **xa,ya,xb,**rp,rn,threshold, *algorithm1*, *algorithm2*, *oversampler*,i=0

1 **While(if** i<N)

2 model_p = *algorithm1* **(xa,ya)**

3 **P, yb =** model_p**(xb)**[*Judge* **yb** *according to* **P** *and threshold*]

4 **(xb_new,yb_newp) = (xb,yb) if** $p_{i\in}(rp,rn)$

5 **if (xb_new,yb_newp)=∅ then break;**

6 **(xb_new_s,yb_new_s) =** *oversampler***(xb_new,yb_newp)**

7 **(xa,ya) =(xa,ya)+ (xb_new_s,yb_new_s)**

8 i = i+1

```
8       End for
10      Model = algrithom2(xa,ya)
```

6.4 迁移算法内容的说明

1) 算法 3 行中，根据自定 threshold 判断是否是正类，因为当类不均衡的时候，少数类很容易被误判为多数类，调节 threshold 是有必要的。
2) 算法 6 行中我们用过抽样器对(**xb_new**,**yb_newp**)过抽样，既保证了算法对于类不均衡的稳定性，又保证了 xb 场景数据加入的比重。
3) 算法 2 行和 10 行，我们分别用算法 algorithm1 去预测概率 algorithm2 去学习数据.algorithm1 是预测概率准确度的算法，algorithm 是 auc 得分高的算法。

6.5 训练数据

我们将 FS，FN，FA，FG，FW 正对 A 集 B 集的特征匹配后，得到源领域知识集 Train_for_FS/FN/FA/FG/FW，并且使用 5 折交叉验证选取了表现最优的 A 模型（学习 A 场景知识最优模型）和其基于的数据集 Train_for_FW.使用它作为源领域知识，以下我们将其称为 TrainA_for_B.使用同样的数据预处理和特征工程我们得到了对应的 B 训练集 Train_B 以及测试集 Test_B.

（**xa**,**ya**）=TrainA_for_B；**xb=**Train_B

6.6 参数说明

6.6.1 rp,rn,threshold

这一部分的阈值我们通过对算法第一轮循环时对 **xb** 的预测概率 **p** 确定.我们可以发现用 0.5 做分正负类阈值时基本没有正类，这在我们先验知识中是不合理的.同时我们可以认识到这是一个非平衡类问题.根据 **p** 分布，我们选取(rp,rn,threshold)=(0.6,0.7,0.62)

6.6.2 algorithm1，algorithm2

algorithm1 目的是达到最准概率，*algorithm2* 目的是达到最高 AUC 我们选取在 A_for_B 集下 5 折交叉测试中 auc 表现最优的 xgb 模型作为 *algorithm2，更*改 xgb 的对概率准确度影响极大的 scale_pos_weight 参数后得到 *algorithm1，即 algorithm1，algorithm2=（xgb_forB1,xgb_forB2）*

6.6.3 oversampler

由于初始 A 场景对 B 场景学习能力有限，容易出错，产生误分类噪声样本;B 场景数据量少，对于学习模型影响不大，影响迁移性能；该问题为类不平衡问题.我们选择了既能删除噪声点，又能增加少数类样本的集成抽样算法: SMOTE + Tomek.

7 代码文件的说明

Feature eng.py

特征工程，运行此文件会载入训练集，寻找特征的最优处理方式，最后将处理方式（flags 文件）保存到 pickles 目录下。

Main.py

生成适合用来训练模型的数据。运行此文件会载入训练集，根据 pickles 目录下保存的 flags 文件，对训练集数据进行处理。最后将处理后的数据保存在 pickles 目录下。

Mymodels.py

针对 A 集的算法模型。

Predict_B_3.py

针对 B 集的算法模型

Preprocess.py

用来预处理的工具库。运行时请更改该文件中 work_path 和 data_path 为合适的路径。

Version3.py

线上部分的代码。载入训练集，载入模型，预测并保存预测结果为指定的格式。

Test.py

训练模型并保存在 pickles 目录下。

本比赛全量代码的运行顺序为：首先，调整 Preprocess.py 文件中的路径。然后，打开 Feature Eng.文件（大约需要 1-2 个小时）。最后，打开 main.py 文件。（会占用约 1G 硬盘空间）。打开 test.py，生成模型并保存在本地。

8 有待改进的地方

首先，由于本次比赛提交次数的限制，我们没有将算法的最优版本上传，所以预测 AUC 的成绩还有继续提高的空间。（最优版本的 B 集算法见文件中的 Pb4.py 文件）

其次，受时间和算力所限，我们所使用的模型没有进行太多的参数调优。如果能有充足的时间来进行调参（parameter tunning）的话，预计还可以将成绩继续提升几个千分位。

还有，我们的迁移算法的实现只是最初步的版本，还没有加入更多实用的功能。B 集的算法也是基于直接的 XGBoost 模型实现的，还没有将我们在 A 集上运用最有效的算法加入进去。