实 验 报 告

实验名称 网贷平台的风控建模与调优

课程名称 网络数据风控技术

专业班级：2021 级数据科学与大数据技术专业

学生姓名：刘博洋　　学　　号：2153538

指导教师：叶晨　　成　　绩：

实验日期：2023-11-5

同济大学

一、 实验背景

信用贷（原蚂蚁借呗）是支付宝平台提供的小额信用贷款服务。互联网小额信用贷款平台（简称网贷平台）根据风险控制和准入标准对支付宝实名用户进行筛选，为筛选出的优质客户提供不同的借款额度。网贷平台的快速发展得益于互联网平台建立了自建的场景流量信用评分体系，形成了获客和贷款的闭环体验。

本实验需要通过约400个数据维度评估用户当前的信用状态，结合脱敏处理过的样本数据，对贷款风险进行建模，预测用户6个月内的逾期率，为平台提供关键的决策依据。

二、 问题分析与数据准备（10%）

项目数据来自于国内网络借贷行业的贷款风险数据，我们所需要预测的目标变量为特定用户的信用违约标签，而把用户的各类特征信息、包括基本信息、网络行为、借款记录等作为因变量。数据集包括30000条训练集和30000条测试集，以GBK格式的csv文件给出。查看数据可以发现，数据集存在较多的重复地理特征，以及众多非数值特征，并且存在较多缺失值；并且由于特征较多，所以需要进行特征工程，将包括地理位置、日期等非数值特征进行处理。随后需要选择不同的模型对数据集进行训练和预测，评估其效果，并采取一定的模型融合方法，使得模型在训练集上能取得更好的效果。

三、 数据清洗与预处理（20%）

（应包含对缺失值的多维度处理、对离群点的剔除、对变量进行归一化处理等内容。）

3.1 缺失值处理

首先可以观察到用户登录信息和用户更新信息没有缺失值，不用进行缺失值处理，然后查看 master 里特征的缺失值占比，可以发现有两个特征 WeblogInfo_1，WeblogInfo_3 的缺失比例接近百分之百，遂直接删除

WeblogInfo_1	0.967667
WeblogInfo_3	0.967667
UserInfo_12	0.630300
UserInfo_13	0.630300
UserInfo_11	0.630300
WeblogInfo_20	0.268333
WeblogInfo_21	0.102467
WeblogInfo_19	0.098767
WeblogInfo_2	0.055267
WeblogInfo_4	0.055033
WeblogInfo_5	0.055033

随后依次处理几个缺失值较高的特征，首先通过画图可可视化各个特征对目标变量的影响UserInfo12，UserInfo13 UserInfo11 的缺失值，统一设置为 2.0，然后将缺失的 webinfo19，20 和 21 设置为不详即可。

然后统一处理其他缺失值的特征，我采取用均值进行填充的方法，对具有缺失值的特征进行填充

并且通过计算各特征的标准差，而标准差接近 0 则代表该特征分布极为均匀，对目标变量几乎造不成任何影响，故直接剔除。

3.2 归一化处理

在处理离群点之前，需要先对非数值特征即文本进行归一化处理。对于名为'UserInfo_9'的列，使用 strip()函数去掉每个字符串值两端的空格；

对于名为'UserupdateInfo1'的列，使用 lower()函数将每个字符串值转换为小写字母，以确保不同的大小写形式被视为相同的值。

然后对城市名进行更加精细化的归一化处理，观察数据可以发现地理数据中是存在如“泰州市”和“泰州”这样的情况的，如果不对其进行处理，后期进行独热编码分类时可能会出现错误分类的情况，所以我们使用 encodingstr(s)函数，去掉城市名末尾的“市”字，而我们又添加的 encodingstrregion(s)函数对省级地理单位进行处理,对于自治区等便把后面的自治区后缀去掉，否则则把“省”字去除。

```
    # 去掉空格
master['UserInfo_9'] = master['UserInfo_9'].apply(lambda x: x.strip())
## 去掉大小写
userupdateinfo['UserupdateInfo1']
=userupdateinfo['UserupdateInfo1'].apply(lambda x:x.lower())
## 将 UserInfo 中城市名归一化
def encodingstr(s):
    regex = re.compile(r'.+市')
    if regex.search(s):
        s = s[:-1]
        return s
    else:
        return s
def encodingregion(s):
    special = ['内蒙古自治区', '宁夏回族自治区', '广西壮族自治区', '新疆维吾尔自治
区']
    ret = ['内蒙古', '宁夏', '广西', '新疆']
    for i in range(len(special)):
        if s == special[i]:
            return ret[i]
    regex1 = re.compile(r'.+省')
    regex2 = re.compile(r'.+市')
    if regex1.search(s):
        s = s[:-1]
        return s
    elif regex2.search(s):
        s = s[:-1]
        return s
    else:
        return s
```

3.2 离群点处理

应用 3 倍标准差原则，对所有的数值型特征计算其标准差，识别出分布在 3 倍标准差之外的数据点，并用特征的均值进行替代。

```
    std_threshold = 3  # 阈值为均值的 3 倍标准差
for col in numeric_cols:
    if col != 'Idx' and col != 'target' and col != 'source' and col != 'ListingInfo':
        mean = master[col].mean()
        std = master[col].std()
        # 识别离群点
        outliers = master[abs(master[col] - mean) > std_threshold * std]
        # 用均值替换离群点
        master.loc[abs(master[col] - mean) > std_threshold * std, col] = mean
```

四、 特征工程（30%）

（应包含地理位置信息、成交时间等特征构建，类别特征编码，组合特征构建，特征选择，类别不平衡处理等。）

4.1 特殊特征处理与构建

可以看到，UserInfo_2,UserInfo_4,UserInfo_7,UserInfo_8,UserInfo_20 都是地理特征中的城市属性，由于城市数量过多，如果按照城市直接进行分类，在独热编码之后会产生非常高维的特征，这样在训练时，较为稀疏的特征分布是很难让模型进行有效的学习的。我们选择按照城市等级，将所有的城市分为一线到四线一共四个类别，通过分类处理将地理特征离散化和数值化，这样可以有效的对分类过于分散的城市特征进行有效的归纳和综合。

```
    first_tier = ['北京', '上海', '广州', '深圳']
new_first_tier = ['成都', '重庆', '杭州', '武汉', '西安', '郑州', '青岛', '长沙', '天津', '苏州', '南京', '东莞', '沈阳', '合肥', '佛山']
second_tier = ['昆明', '福州', '无锡', '厦门', '哈尔滨', '长春', '南昌', '济南', '宁波', '大连', '贵阳', '温州', '石家庄', '泉州', '南宁', '金华', '常州', '珠海', '惠州', '嘉兴', '南通', '中山', '保定', '兰州', '台州', '徐州', '太原', '绍兴', '烟台', '廊坊']
```

```
third_tier = ['海口', '汕头', '潍坛', '扬州', '洛阳', '乌鲁木齐', '临沂', '唐山', '镇江', '盐城', '湖州', '赣州', '漳州', '揭阳', '江门', '桂林', '邯郸', '泰州', '济宁', '呼和浩特', '咸阳', '芜湖', '三亚', '阜阳','淮安', '遵义', '银川', '衡阳', '上饶', '柳州', '淄博', '莆田', '绵阳', '湛江', '商丘', '宜昌','沧州', '连云港', '南阳', '蚌埠', '驻马店', '滁州', '邢台', '潮州', '秦皇岛', '肇庆', '荆州', '周口', '马鞍山', '清远', '宿州', '威海', '九江', '新乡', '信阳', '襄阳', '岳阳', '安庆', '菏泽', '宜春', '黄冈', '泰安', '宿迁', '株洲', '宁德', '鞍山', '南充', '六安', '大庆', '舟山']
fourth_tier = ['常德', '渭南', '孝感', '丽水', '运城', '德州', '张家口', '鄂尔多斯', '阳江', '泸州', '丹东', '曲靖', '乐山', '许昌', '湘潭', '晋中', '安阳', '齐齐哈尔', '北海', '宝鸡', '抚州', '景德镇', '延安', '三明', '抚顺', '亳州', '日照', '西宁', '衢州', '拉萨', '淮北', '焦作', '平顶山', '滨州', '吉安', '濮阳', '眉山', '池州', '荆门', '铜仁', '长治', '衡水', '铜陵', '承德', '达州', '邵阳', '德阳', '龙岩', '南平', '淮南', '黄石', '营口', '东营', '吉林', '韶关', '枣庄', '包头', '怀化', '宣城', '临汾', '聊城', '梅州', '盘锦', '锦州', '榆林', '玉林', '十堰', '汕尾', '咸宁', '宜宾', '永州', '益阳','黔南州', '黔东南', '恩施', '红河', '大理', '大同', '鄂州', '忻州', '吕梁', '黄山', '开封', '郴州', '茂名', '漯河', '葫芦岛', '河源', '娄底', '延边']
```

对于省份特征，我们可以看到 UserInfo_7 和 UserInfo_19 是省份信息，由于省份特征不像城市那么分散，所以我们选择违约率超过一个阈值（我这里设置为 0.05）的省份或直辖市构造二值特征将特征进行数值化，取值为 0 或 1 。

其次我们还需要处理表中成交时间这一特征，为了把日期数据转化为数值型特征并进行离散化，我们考虑将日期中比较有代表性的日和月特征提取出来成为单独两个特征，并删除原有的日期特征。

随后我们提取 userupdate 表中的特征，加入 master 训练集，本次项目我没有考虑和处理 loginfo 表中的，特征，因为从现实世界来说，可以认为用户的登录记录与其是否违约并不存在什么关联，因此选择直接放弃这部分数据，以免产生过多噪声。

4.2 类别特征编码

采用独热编码对已经处理好的数据集进行展开，将每个类别转化为一个独立的二进制特征，因此能够完整地保留原始类别信息。这有助于

模型识别和理解不同的类别，并且能够一定程度上处理确实数据，提升模型性能。最后一共分解出5443维的特征，随后对模型数据进行scaling，有利于模型对特征进行更好的学习和训练。

4.3特征选择以及不平衡处理

对于特征的选择以及可能存在分类不平衡的处理，我这里直接考虑通过xgboost和lighgbm内置的特征选择feature importance来实现，并且可以在模型中通过设置scale_pos_weight和is_unbalance 参数来自动平衡正负样本权重，这对于二分类问题十分有效。对于特征组合问题，

五、建模与优化（30%）

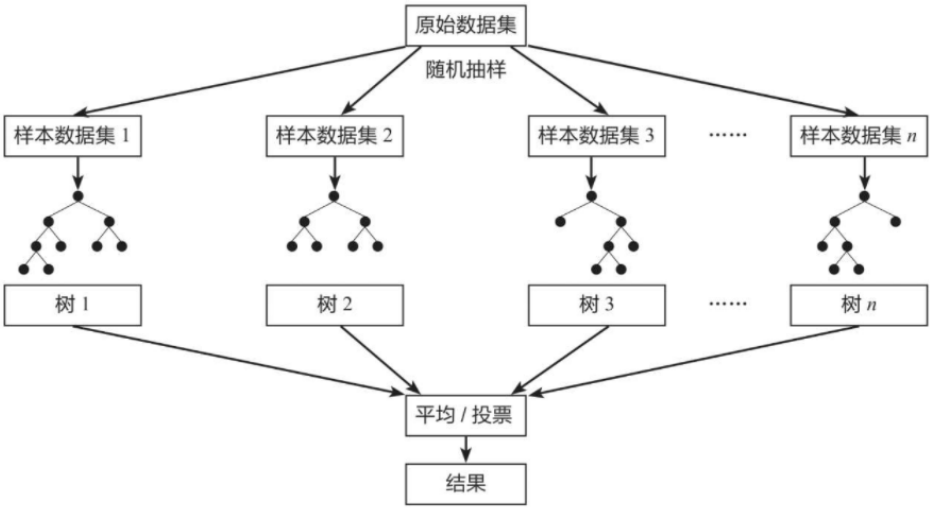
（应包含逻辑回归模型、xgboost或lightgbm等机器学习模型，并采用模型融合方法提高模型效果）

5.1 逻辑回归

逻辑回归是处理二分类问题的最经典的机器学习方法，我首先尝试用最简单的逻辑回归对数据集进行训练和预测。但是可以预见的是，简单的逻辑回归在处理如此高维的数据时会遇到许多困难，高维度数据通常包含大量特征，其中许多可能对问题的解释力较小，而逻辑回归并没有进行特征选择，而是接纳所有特征，这可能会导致无关特征被引入，降低模型性能，并且逻辑回归泛化能力不强，结果也比较符合预期，虽然在训练集上的准确率达到了92%，但是auc分数只有70，并且在提交平台上进行预测的分数只有69.6分。

5.2 random Forest模型

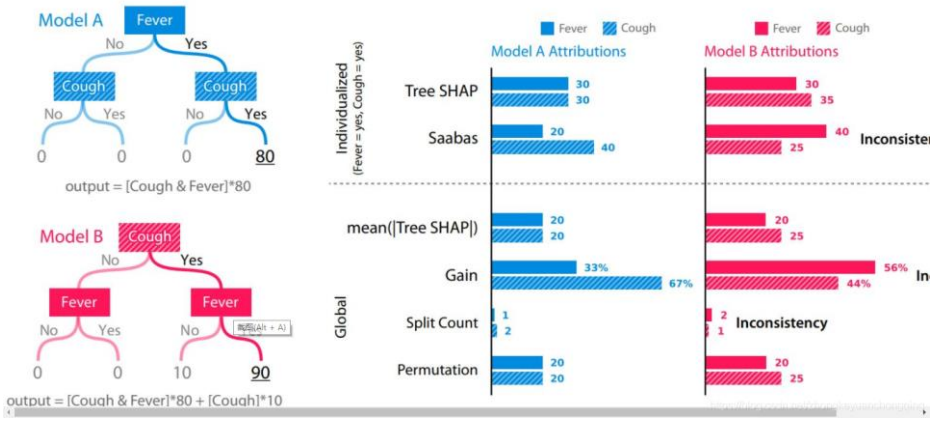
随机森林指的是利用多棵树对样本进行训练并预测的一种分类器，它利用 bagging 的思想，它由多个决策树组成，每棵树都是独立训练的。随机森林通过对每棵树的预测结果进行平均或投票来进行预测，从而提高了模型的鲁棒性和准确性。但是很多情况下，当数据量比较大或者维度比较高的情况下，生成的决策树数量太多容易产生过拟合，在训练集上表现一般，并且需要耗费较长的训练时间。



我使用随机森林模型进行训练，并且已经控制了 min_samples_split 和 min_samples_leaf 防止过拟合，可是仍然在训练集上得到的准确率为 0.9267，而 AUC Score(Train)甚至达到了 0.999998，说明还是发生了严重的过拟合。在平台训练集上的分数也只有 51.6。

5.3 xgboost 模型

Xgboost 是一种强大的梯度提升树算法，广泛用于机器学习和实际应用中。它通过构建多个弱预测模型的集合来提升整体预测能力。每棵树都是在前一棵树的残差基础上构建的。我们使用 Boosting 算法来不断迭代，增加弱分类器并根据一定的学习率进行学习，提高模型精度。



其次，由于 XGBoost 使用了正则化项来控制模型的复杂度，并采用了特殊的梯度下降方法进行训练，使得其在处理高维稀疏数据时，具有较好的效果。

并且在本项目中，可以利用 xgboost 进行方便的特征选择，选择重要度 topN 的特征。Xgboost 库提供了方便的模型参数设置，这里我选择的参数为

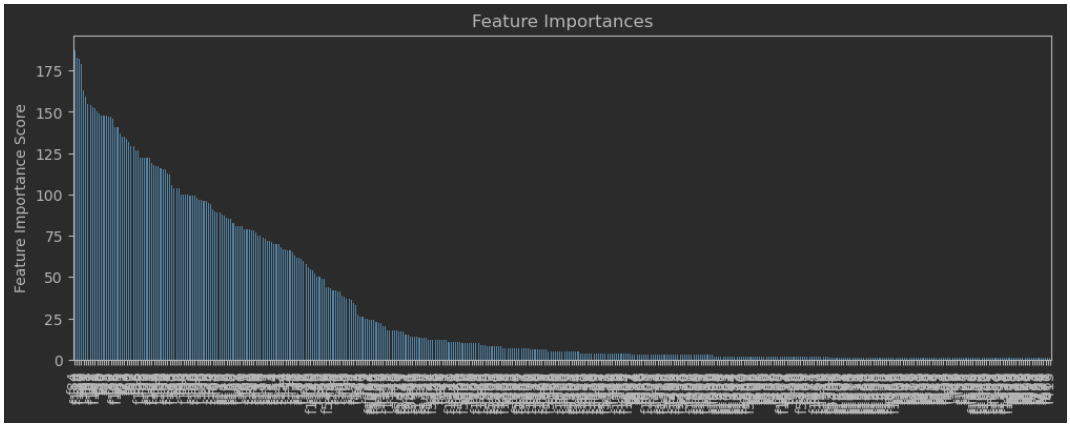
```
    xgb1 = XGBClassifier(
    learning_rate=0.04,
    n_estimators=200,
    max_depth=7,
    min_child_weight=1.0,
    gamma=0,
    subsample=0.8,
    colsample_bytree=0.8,
    objective='binary:logistic',
    nthread=4,
    scale_pos_weight=3,
    seed=27
)
```

其中学习率不用设的太大，为了模型的精度，但也不能设置太小，因为防止模型过拟合，而 n_estimator 是弱分类器的数量，并且我们通过设置 scale_pos_weight 对类别不平衡的问题进行处理。

最后我们得到模型在训练集上的结果为

准确率 ： 0.9408

AUC 得分（训练集）：0.981709

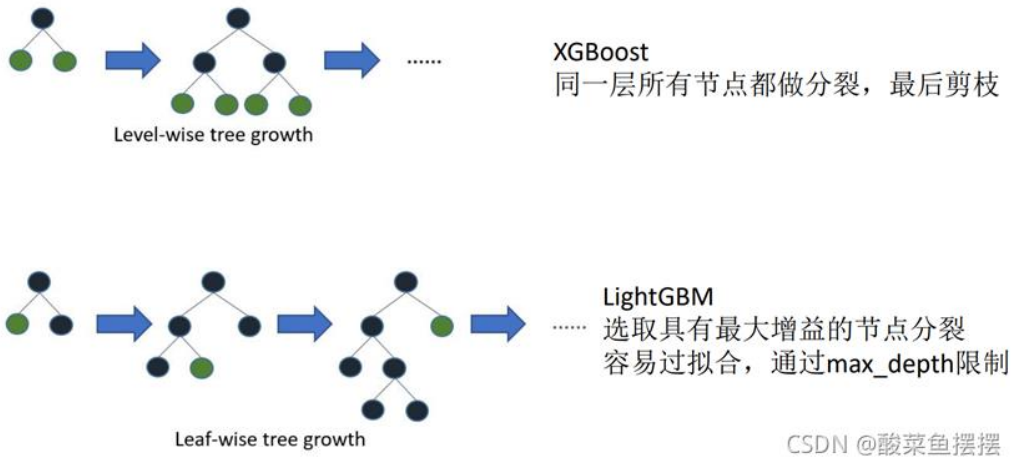


模型同样进行的特征选择，按照重要度分数对特征进行排序，通过多次调参，最后在平台验证集上的最好分数为 75.42，已经达到较好的效果。

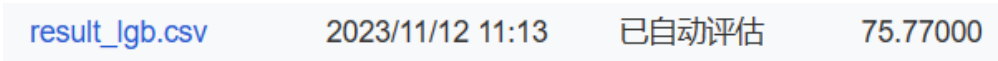
result.csv	2023/11/06 19:44	已自动评估	75.42000

5.4 lightgbm 模型

而 LightGBM 与 xgboost 一样，都是一个实现 GBDT 梯度提升树算法的框架，但是相比于 xgboost，lightgbm 选择了基于 histogram 的决策树算法，每次选取最大增益的节点进行分裂，具有更快的训练速度、更低的内存消耗、更好的准确率等优点，本模型也比较适合采用 lightgbm 进行训练。



为了防止过拟合，我在模型中引入 L1 和 L2 正则化惩罚项，并将树的层数设置的较深，叶子节点没有设置的太高，最中经过多次调优，可以得到在训练集上模型的 auc 分数达到了 0.979，而在平台测试集上的最好分数已经达到了很高的 75.77




```
    params = {
    'objective': 'binary',
    'boosting_type': 'gbdt',
    'num_leaves': 40,
    'learning_rate': 0.02,
    'feature_fraction': 0.2,
    'max_depth':11,
    'reg_alpha':0.4,
    'reg_lambda':0.4,
    'n_estimators':500,
}
```

以上是取得最好分数的 lightgbm 参数

5.5 catboost 模型

通过学习课堂上高分同学的分享，我又尝试了 catboost 方法。它在处理分类问题时，可以自动处理类别特征，无需手动进行特征编码。CatBoost 的原理与 XGBoost 和 LightGBM 类似，同样是通过将多个弱学习器组合成一个强学习器。不同之处在于，CatBoost 使用了一种新的损失函数，即加权交叉熵损失函数，可以有效地处理类别不平衡问题。

CatBoost 使用了一种称为 Ordered Boosting 的特征选择算法，并且在处理分类问题时，它使用了一种称为 Target Encoding 的方法，可将类别特征转化为一组实数值，以自动处理类别特征。

最后，CatBoost 还使用了基于对称树的决策树算法，使得在处理高维稀疏数据时，具有较好的效果。

在训练集上，通过多次调参，选取了一个比较好的结果，模型在迭代 1400 余次时达到了最小损失，准确率:为 0.9245，AUC 分数为 0.7625820673638272，但是可能发生了过拟合，在平台测试集上的最好分数仅为 74.72，没有达到预期好的效果。

result_ctb.csv	2023/11/12 12:21	已自动评估	74.72000

5.6 模型融合

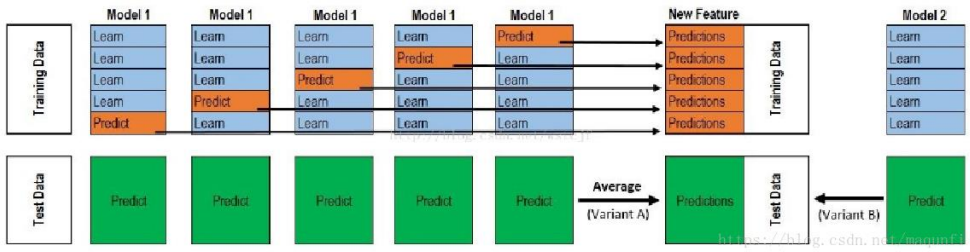
1.voting 投票法

投票法是模型融合算法中最为简单的，它对所有评估器输出的结果按类别进行计数，出现次数最多的类别就是融合模型输出的类别。

我最开始尝试了投票法对随机森林、xgboost、lightgbm 和 catboost 进行模型融合，但是并没有对不同的分类器按照效果赋予不同权重，最后得出融合后的准确率：0.924，AUC 分数：0.7636958387300533，效果十分不错，在平台验证集上的分数也有 75.26。

result_merge.csv	2023/11/12 22:49	已自动评估	75.26000

2.stacking



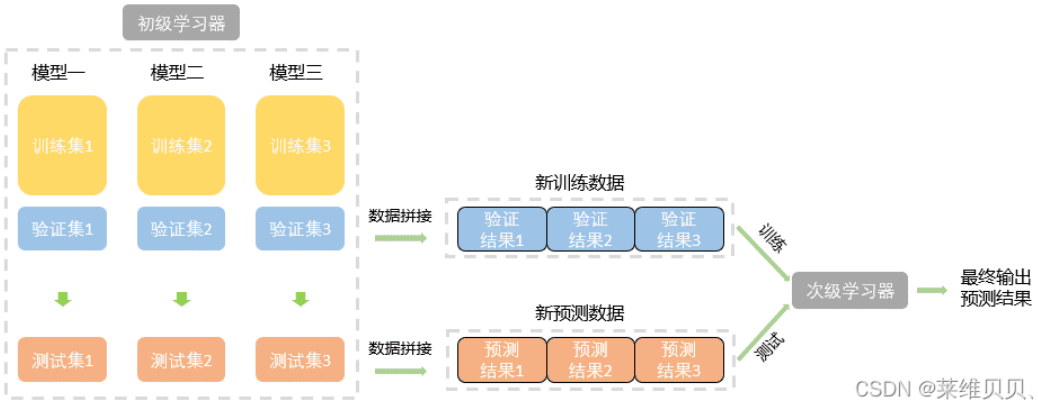
Stacking 的主要思想是训练模型来学习使用底层学习器的预测结果，基模型在所有数据集上生成预测结果，次学习器会基于模型的预测结果进行再训练，并且用到了 K 折交叉进行验证。我尝试将效果较好的随机森林、xgboost 和 lightgbm、catboost 作为底层学习器，进行

stacking 融合，最后得到训练集上的准确率：0.9245

AUC 分数：0.7560697149893767，在平台验证集上的分数为 75.02,，虽然没有得到明显提升，但是也达到了满分的标准，说明模型融合的效果还不错。

result_merge.csv	2023/11/12 21:14	已自动评估	75.02000

3.blending



blending 在 stacking 的基础上加了划分数据，划分之后的训练数据一部分训练基模型，一部分经模型预测后作为新的特征训练元模型。测试数据同样经过基模型预测，形成新的测试数据。最后，元模型对新的测试数据进行预测。Blending 方法最后我这里得到的测试结果不如 stacking，可能需要长时间的调参和调整每一层的模型才能获得较好的结果。

六、 结论和心得体会（8%）

（实验结果，并对实验现象及处理方法、实验中存在的问题等进行分析和讨论，对实验的进一步想法或改进意见。）

本次实验是对较大规模信贷和风控数据的一次较为复杂的建模，在这次项目中，我比较全面的了解和实践了风控建模的全过程，从数据预

处理、数据清洗、再到特征工程、特征提取与分析，再到最后的模型构建与参数调优，我不仅在实践中巩固了数据处理和特征工程的相关方法，还使用了一些功能强大的模型，也从其他同学的模型中进行借鉴与学习，最终也是取得了平台上 75.77 分的不错的成绩。

在数据预处理以及特征工程方面，我属于比较按部就班，也比较迅速和顺利，按照标准化的步骤对有关噪声以及离群点进行处理，并统一进行独热编码处理，得到了质量较高的数据集。

对于最后模型的选择，我花费了很多时间，也逐渐领悟了各个模型的优缺点。其中，XGBoost 在传统机器学习领域仍然是最常用的算法之一，特别是在结构化数据的分类、回归和排序任务中表现突出。LightGBM 在大规模数据集和高维度数据上表现更佳，适用于处理文本分类、图像分类、推荐系统等领域的数据。CatBoost 在处理类别特征和缺失值方面表现出色，适用于电商推荐、医疗预测、金融风控等领域的数据。其实模型构建过程中调参的过程是比较痛苦又充满期待的过程，一方面是要根据数据的特点以及算法的特点进行选择，另一方面也需要不断地尝试和等待，在一次次的测试分数的提高中我不仅收获了较大的成就感，也取得了许多有效的调参经验。

一开始其实并没有考虑将模型进行融合，后来通过同学的分享了解到了比较常用的投票、stacking 和 blending 等模型融合方法，并逐一进行尝试，在多次调参后发现可能由于融合后的模型产生了比较大的过拟合，对训练集的效果并不好，遂没有再进行过多深入的钻研。

七、 参考文献：（2%）

[1]于晓虹，楼文高. 基于随机森林的P2P网贷信用风险评价、预警与实证研究[J]. 金融理论与实践，2016(2):6.

[2]Boosting三巨头：XGBoost、LightGBM和CatBoost（发展、原理、区别和联系，附代码和案例
(https://blog.csdn.net/qq_41667743/article/details/129417794)

[3]【集成学习】Blending和Stacking
(https://blog.csdn.net/a8689756/article/details/115614283)