復旦大學

《机器学习》研究报告

信用卡交易欺诈检测

目录

一、 研究背景

2021 年，与信用卡欺诈相关的损失超过 120 亿美元，同比增长近 11%。就重大财务损失、信任和信誉而言，这是银行、客户和商户面临的一个令人担忧的问题。

电子商务相关欺诈一直在以约 13% 的复合年增长率 (CAGR) 增加。由于欺诈性信用卡交易急剧增加，在交易时检测欺诈行为对于帮助消费者和银行非常重要。机器学习可以通过训练信用卡交易模型，然后使用这些模型更快、更准确地检测欺诈交易，在预测欺诈方面发挥至关重要的作用。

故在本次研究中，我们拟参考英特尔的类似实现方案，基于英特尔提供的信用卡交易数据，训练多个机器学习模型，有效预测信用卡交易是否为欺诈交易。并借助英特尔® oneAPI AI 分析工具包中数据分析和机器学习相关的优化库，跟官方的机器学习库进行性能对比分析。

二、 数据预处理

数据介绍

该数据集包含欧洲持卡人 2013 年 9 月通过信用卡进行的交易。该数据集显示了两天内发生的交易，其中 284,807 笔交易中有 492 笔欺诈。数据集高度不平衡，正类（欺诈）占所有交易的 0.172%。它仅包含 PCA 变换结果的数字输入变量。特征 V1、V2、…V28 是通过 PCA 获得的主要成分，唯一未通过 PCA 转换的特征是“时间”和“金额”。特征“时间”包含数据集中每个事务与第一个事务之间经过的秒数。特征“金额”是交易金额，该特征可用于示例相关的成本敏感学习。特征“类别”是响应变量，如果存在欺诈，则取值 1，否则取值 0。考虑到类别不平衡率，我们拟打算使用精确率-召回率曲线下面积 (AUPRC) 来测量准确度。混淆矩阵精度对于不平衡分类没有意义。

数据清洗

观察数据集发现，该数据集中不存在全空行或者全空列。检查重复行的时候发现，该数据集中存在共 1081 行重复行，采用 drop_duplicates(keep='first')处理后剩余 283726 行。

特征工程

考虑到数据集中的变量仅包含 PCA 变换结果的数字输入变量，故暂时不对其进行其他的特征工程处理，例如特征选择和特征衍生等操作。

鉴于数据介绍部分所提到的数据集存在高度不平衡的现象，在后续机器学习模型的采样过程中，我们会利用 SMOTE 函数进行平衡采样，并和不平衡采样直接进行训练的模型进行对比，以期获得更好的效果。

训练集、测试集划分

针对预处理过后的数据集，我们对其进行训练集和测试集合的划分，其中训练集占比30%，测试机占比 70%。

三、 模型训练

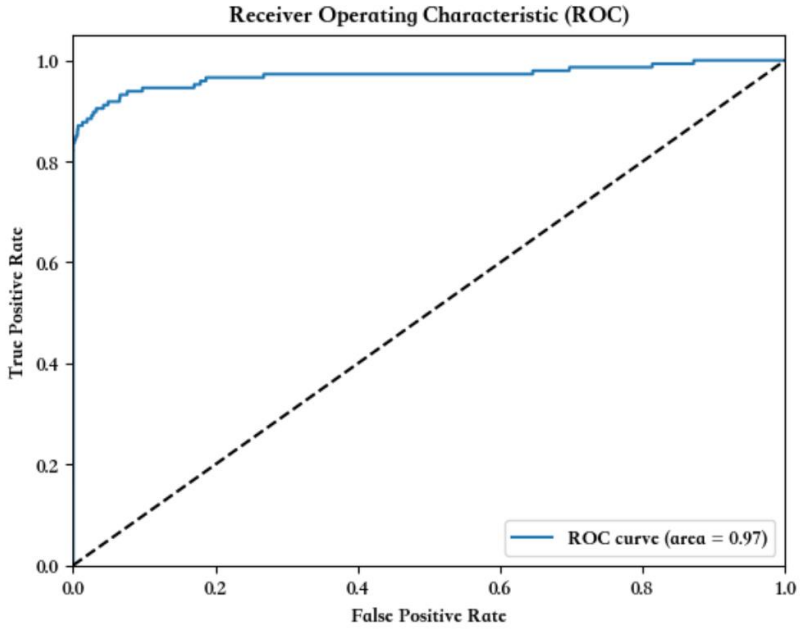
随机森林模型

无采样

我们首先选择使用随机森林模型对训练集进行训练，并在训练过程中使用网格搜索以获得使结果最优的参数组合，其中最优参数组合为：该随机森林模型中决策树的数量为 50(n_estimators=50)，在模型性能提高和训练时间内存损耗之间取得平衡；决策树的最大深度为 10(max_depth=10)，以此来控制树的复杂度和过拟合的风险;进行拆分所需的最小样本数为 2(min_samples_split=2)，在模型过拟合和欠拟合之间取得平衡；每个决策树考虑的特征数量为 3(max_features=3)；模型训练过程中要并行运行的作业数量为 2（n_jobs=2）。

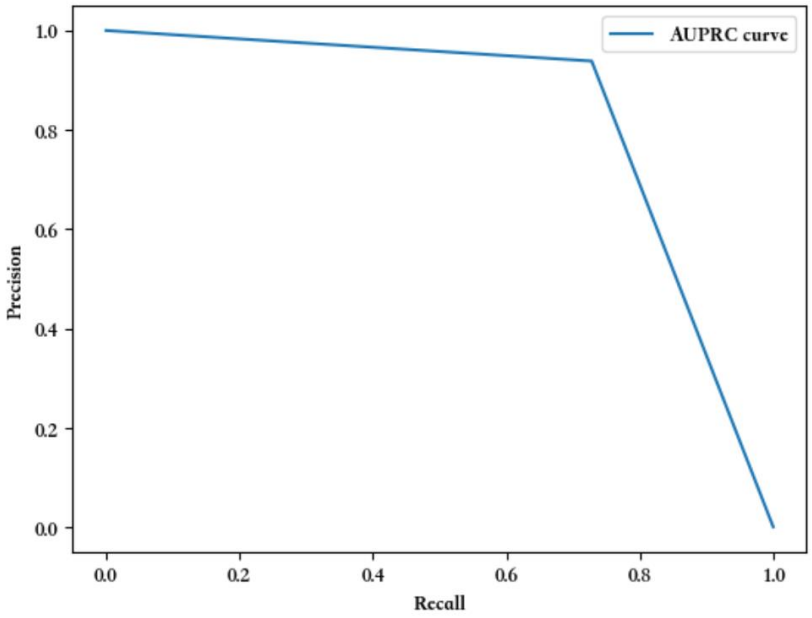
在上述参数基础上，我们得出了以下模型拟合结果，accuracy 为 0.999，precision 为 0.938，recall 为 0.727，roc_auc 为 0.970，f1 为 0.819，ROC 曲线如下图 1 所示：

图 1 无采样随机森林模型 ROC 图



为更好的描述不平衡分类问题的模型拟合效果，我们进一步使用精确率-召回率曲线下面积 (AUPRC)衡量模型效果，其值达到 0.683，结果如下图 2 所示：

图 2 无采样随机森林模型 AUPRC 图



我们认为模型整体拟合效果较为优异。

有采样

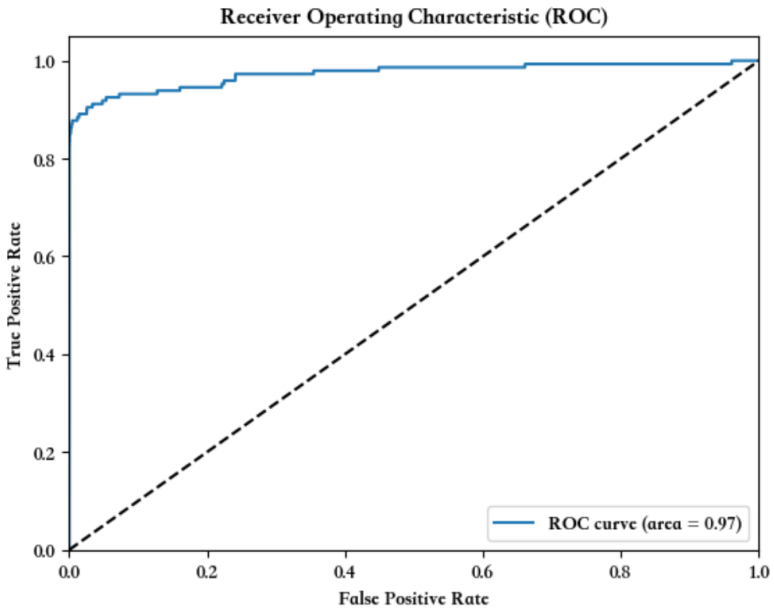
第二部分数据预处理中表明该样本存在高度不平衡的情况，故我们对其进行平衡采样才处理。

首先，我们创建了一个 SMOTE 实例，通过 sampling_strategy=0.01 指定了过采样的目标类别样本比例。接下来，创建了一个 RandomUnderSampler 实例，同样通过 sampling_strategy=0.01 指定了欠采样的目标类别样本比例。然后，通过 Pipeline 创建了一个数据处理的流程。流程中包含两个步骤，第一个步骤是使用 over_sampler 进行过采样，第二个步骤是使用 under_sampler 进行欠采样。最后，调用 fit_resample 方法，将原始的 train_feature（训练特征）和 train_target（训练目标）作为输入，对其进行过采样和欠采样处理。

经过调试，最优比例如下：第一轮对正类样本过采样，使得生成后总正类数量与所有负类数量比例确定为 0.01，这里的 0.01 表示采样后的目标类别样本数量将是原始样本数量的 1%。第二轮对负类样本前欠采样时保持正类数量不变，使得正类数量与负类数量比例确定为 0.01，这里的 0.01 表示采样后的目标类别样本数量将是原始样本数量的 1%。注意这里的所有采样操作均对训练集进行。采样后，训练的目标集一共有 200990 个样本，其中值为 1 的样本有 1990 个。

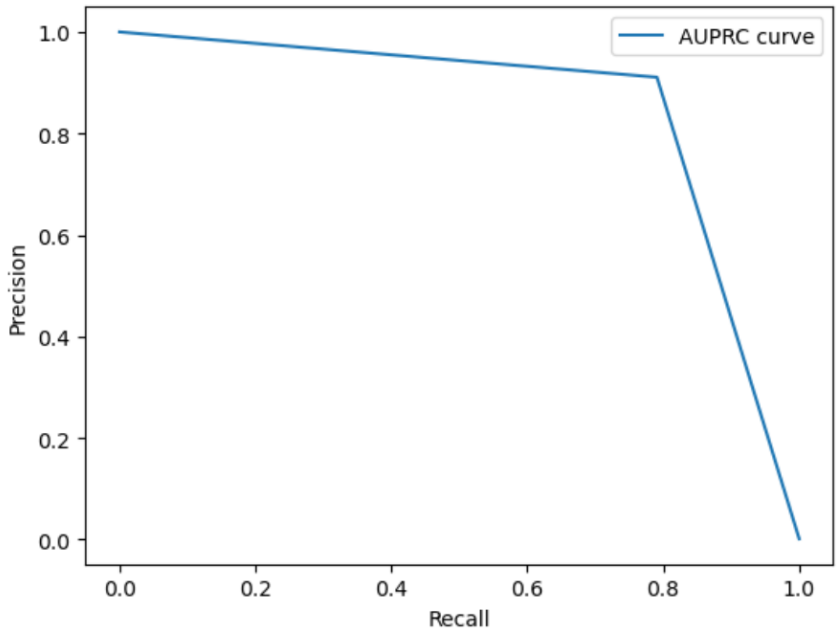
我们沿用上述无采样过程中的最优组合参数，对平衡采样后的数据进行随机森林模型训练，得到了如下模型拟合结果，accuracy 为 0.999，precision 为 0.936，recall 为 0.795，roc_auc 为 0.973，f1 为 0.860，ROC 曲线如下图 2 所示：

图 3 有采样随机森林模型 ROC 图



该模型的 AURPC 值达到 0.745，相比采用未采样训练的随机森林模型取得较大提升，其结果如下图 4 所示：

图 4 有采样随机森林模型 AUPRC 图



纵向比较，相比上述无采样的随机森林模型训练结果，除去精准率以外，我们发现其余所有的评价指标均获得了一定程度上的提升，因而获得拟合效果更好的随机森林模型。由此，我们也可以看出针对不平衡数据集，进行平衡采样的必要性。同时，平衡采样也会在模型拟合更有的情况下，导致个别评价指标出现异常变化情况。

XGBoost 模型

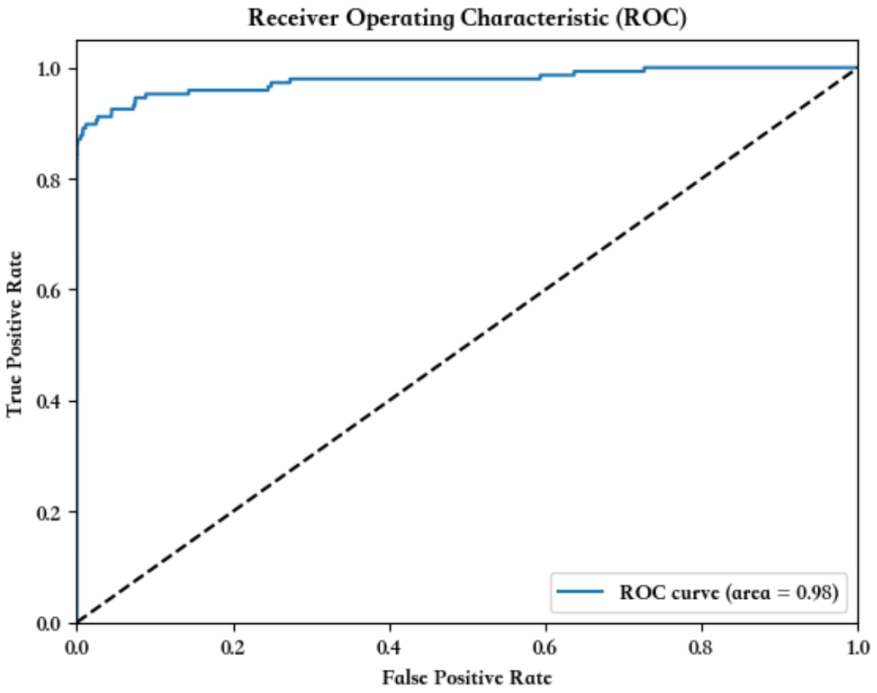
鉴于 XGBoost 模型在以往实践过程中，往往会有比较不错的效果，所以在这里我们选择使用 XGBoost 模型和随机森林模型进行对比。

无采样

我们使用 XGBoost 模型对训练集进行训练，并在训练过程中使用网格搜索以获得使结果最优的参数组合，其中最优参数组合为该 XGBoost 模型中弱分类器决策树的数量为 150(n_estimators=150)，在模型性能提高和训练时间内存损耗之间取得平衡；决策树的最大深度为 9(max_depth=9)，以此来控制树的复杂度和过拟合的风险;学习率设置 0.1 (learning_rate=0.1)，较小的学习率可以使模型更加稳定，但需要更多的弱学习器；样本抽样比例为 1(subsample=1)，表示使用全部样本；特征抽样比例设置为 1(colsample_bytree=1)，表示使用全部特征；叶节点分裂所需的最小损失减少量设置为 0.1(gamma=0.1)，其较大的值可以防止过拟合，控制树的生长。L1 正则化项的权重设置为 0，L2 正则化项的权重设置为 1 (reg_alpha=0，reg_lambda=1),表示这里并没有 L1 正则化项,但进行了 L2 正则化项；模型的优化目标设置为 "binary:logistic"，我们拟使用二分类逻辑回归作为目标。

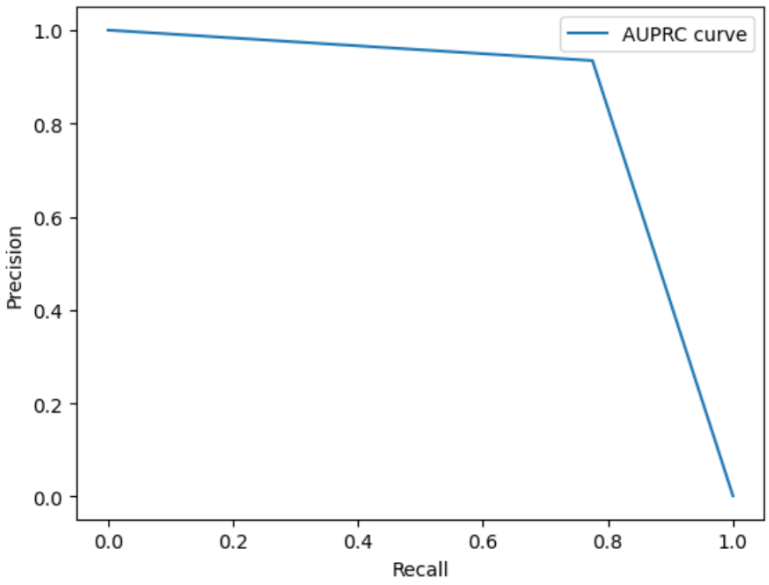
在上述参数基础上，我们得出了以下模型拟合结果，accuracy 为 0.999，precision 为 0.950，recall 为 0.782，roc_auc 为 0.977，f1 为 0.858，ROC 曲线如下图 5 所示：

图 5：无采样 XGBoost 模型 ROC 图



XGBoost 模型的 AURPC 值达到 0.743,其效果如下图 6 所示：

图 6 无采样 XGBoost 模型 AUPRC 图

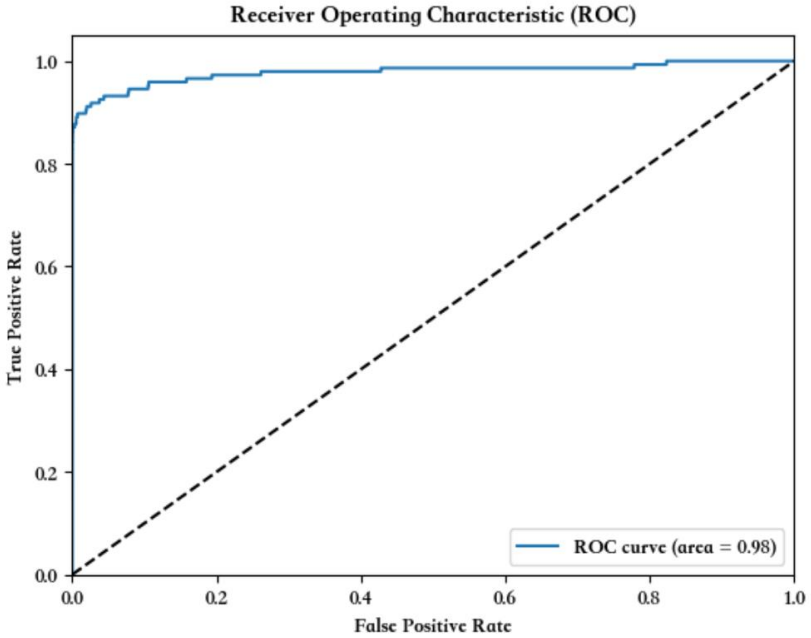


横向比较，相比之前无采样的随机森林模型，XGBoost 模型效果有着极为明显的提升。其中 AUPRC 尤为明显，从 0.683 提升达到了 0.743。

有采样

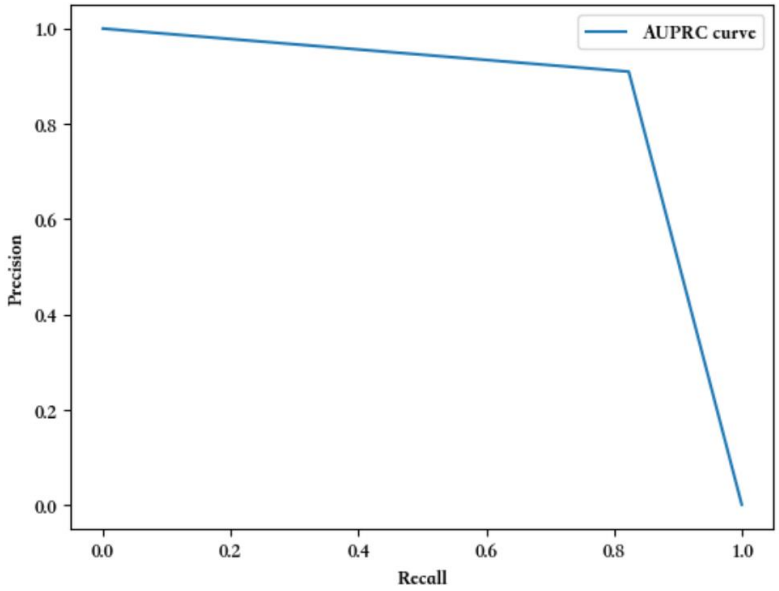
参考随机森林中的操作，我们进行了相同的过采样+欠采样的平衡采样方法，并沿用上述无采样过程中的最优组合参数，对平衡采样后的数据进行 XGBoost 模型训练，得到了如下模型拟合结果，accuracy 为 0.999，precision 为 0.909，recall 为 0.823，roc_auc 为 0.975，f1 为 0.864，ROC 曲线如下图 7 所示：

图 7 有采样 XGBoost 模型 AUPRC 图



其中 AUPRC 为 0.749，其结果如下图 8 所示：

图 8：无采样 XGBoost 模型 AUPRC 图

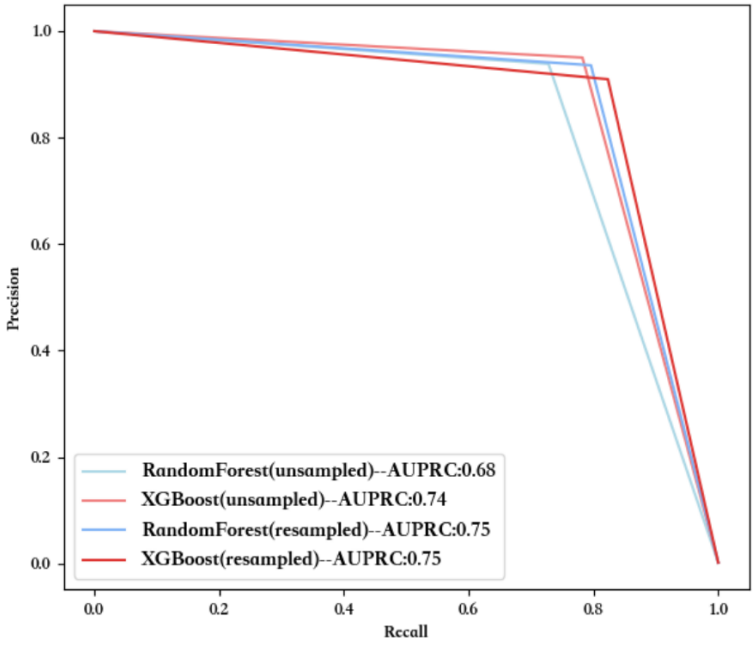


纵向对比发现，我们发现除去精准率有略微降低以外，其余评价指标均获得了一定程度上的提升，因而获得拟合效果更好的 XGBoost 模型。进一步验证了针对不平衡数据集，进行平衡采样的必要性。但重新采样一定程度上也有可能会影响到部分评价指标的改变。

结果对比

横向对比发现，XGBoost 模型的效果总体优于随机森林模型的效果；纵向对比发现，经过平衡采样后，无论是随机森林模型还是 XGBoost 模型均有了较明显的提升，如下图 9 所示：

图 9 本地训练后的模型 AUPRC 结果对比



四、 INTEL ONEAPI 使用对比

我们通过调用 modin 库，使用 modin.python 和 modin.config 来优化提升我们的算法。

速度效率对比

我们可以发现在借用 intel OneAPI 后，模型的训练时间得到了大幅提升。以下是我们做出的在效率上的具体对比。

随机森林模型

对于随机森林模型，我们使用的网格优化参数如下：
'min_samples_split': [2,6],
'n_estimators' :[50,100,150],
'max_depth': [9, 10,13],
'max_features': [3,6,8]

其在本地运行时间超过 30 分钟，在 intel 云上运行时间为 52.83s。最优参数为 max_depth = 10, max_features = 3, min_samples_split = 2, n_estimators = 50。相同训练集，在本地训练时间为 120.45 秒，在 intel 云上训练时间为 0.86 秒。

XGBoost 模型

对于 XGBoost 模型，我们使用的网格优化参数如下：
'colsample_bytree' : [0.8,1],
'gamma' : [0,0.1],
'learning_rate' : [0.1,0.2],
'max_depth' : [9,12],
'min_child_weight' : [3, 5],
'n_estimators' : [150,300],
'reg_alpha' : [0,0.1],
'subsample' : [0.8,1]
‘reg_alpha’:[0,0.1]
‘reg_lambda’:[0,1]

其在本地运行时间超过 1 小时，在 intel 云上运行时间运行时间不足 3 分钟。最优参数为 max_depth = 9,learning_rate = 0.1,n_estimators = 150,subsample = 1,colsample_bytree = 1,gamma = 0.1,reg_alpha = 0,reg_lambda = 1,相同训练集，在本地训练时间为 4.3 秒，在 intel 云上训练时间为 0.61 秒。

模型效果对比

我们发现，在使用 OneAPI 进行机器学习模型训练时，模型效果整体都得到了极大提升，无论是横向随机森林模型与 XGBoost 模型，还是纵向有采样与无采样，模型拟合效果都取得了巨大突破。

针对随机森林模型无采样，原先效果为 accuracy 为 0.999，precision 为 0.938，recall 为 0.727，roc_auc 为 0.970，f1 为 0.819；经过 OneAPI 训练后，模型提升为 accuracy 为 0.999，precision 为 1.0，recall 为 0.771，roc_auc 为 0.953，f1 为 0.871，AUPRC 为 0.772，除 roc_auc 外，所有指标均获得较明显的提升。又考虑到在非平衡数据集中，roc_auc 指标相对不够精准，故认为模型得到全面提升。

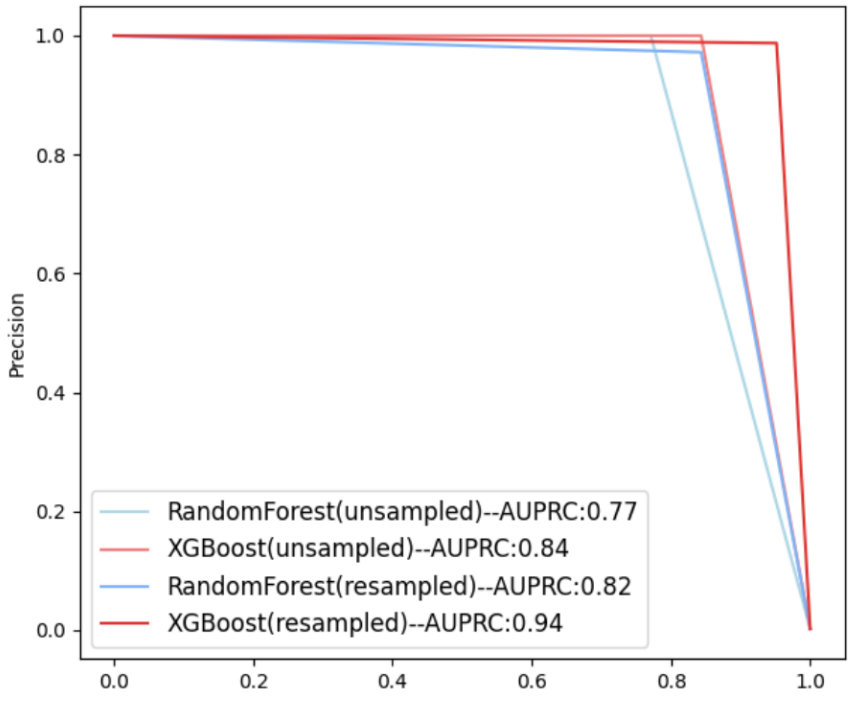
针对随机森林模型有采样，原先效果为 accuracy 为 0.999，precision 为 0.944，recall 为 0.810，roc_auc 为 0.978，f1 为 0.871；经过 OneAPI 训练后，模型提升为 accuracy 为 0.999，precision 为 0.972，recall 为 0.843，roc_auc 为 0.986，f1 为 0.903，AUPRC 为 0.820，模型提升相对于无采样，提升更为明显。

针对 Xgboost 模型无采样，原先效果为 accuracy 为 0.999，precision 为 0.950，recall 为 0.782，roc_auc 为 0.977，f1 为 0.858；经过 OneAPI 训练后，模型提升为 accuracy 为 0.999，precision 为 1.0，recall 为 0.843，roc_auc 为 0.973，f1 为 0.915，AUPRC 为 0.843，模型进一步得到明显提升。

针对 Xgboost 模型有采样，原先效果为 accuracy 为 0.999，precision 为 0.909，recall 为 0.823，roc_auc 为 0.978，f1 为 0.864；经过 OneAPI 训练后，模型提升为 accuracy 为 0.999，precision 为 0.987，recall 为 0.951，roc_auc 为 0.997，f1 为 0.969，AUPRC 为 0.940。

我们发现，通过 OneAPI 的训练，有采样的 XGBoost 模型获得了远优于其他情况的最有效果，符合我们最初的直观认知。4 种不同情形下的模型对比如下图 10 所示：

图 10 经过 OneAPI 训练后的模型 AUPRC 结果对比



下表 1，表 2 中，我们仅分析不同情况下的 AUPRC：

表 1. 无采样情况下本地结果与 OneAPI 结果的 AUPRC 对比

无采样	本地结果	OneAPI 结果
随机森林	0.68	0.77
XGBoost	0.74	0.84

表 2. 有采样情况下本地结果与 OneAPI 结果的 AUPRC 对比

有采样	本地结果	OneAPI 结果
随机森林	0.75	0.82
XGBoost	0.75	0.94

通过上述对比可以明显看到：

（1）在加入采样机制后，模型性能较之前相比均得到一定提高，其中在本地的 XGBoost 提升效果不显著，在 OneAPI 上各模型的提高效果较为明显。

（2）在 OneAPI 上训练，模型性能均较大程度优于在本地的训练结果