Santander Customer Transaction Prediction

摘要：本文主要是做 Santander 银行的客户交易行为预测。桑坦德银行（Santander）又叫西班牙国际银行，是西班牙最大、欧洲第二大银行。Kaggle 竞赛平台上的 Santander Customer Transaction Prediction 这一比赛是让参赛者通过二十万条交易数据作训练集来预测测试集中的哪些数据可能会进行交易。本文通过对数据集进行预处理之后，利用梯度提升模型进行训练来对测试集数据进行预测。之后通过深入挖掘数据，发现数据中掺杂了疑似人造的假数据，对该部分数据深入分析处理之后，最终使得 AUC 值有了较大的提升，提交到 Kaggle 的分数也有了较大提升，取得了排名前 1%的成绩。

1. 问题介绍

桑坦德银行（Santander）又叫西班牙国际银行，是西班牙最大、欧洲第二大银行。Kaggle[1]竞赛平台上的 Santander Customer Transaction Prediction 这个比赛主要是预测哪些客户将来会进行一个特定的交易（无论交易金额多少）。该比赛是个二分类问题，其中 0 表示该客户不会有交易，1 表示该客户会有交易。该比赛的评估标准是 AUC 值，即 ROC[2] 曲线下的面积。

训练数据集包含一个“ID_code”属性（值为 string 类型）、一个“target”（值为 0 或 1）属性以及 200 个特征（从“var_0”到“var_199”均为数值类型）。训练数据集中共有 20 万条数据样本，测试集也有 20 万条数据样本（测试集没有 “target” 属性，需要去预测这个属性值）。训练数据集和测试数据集中的 200 个特征全部进行了脱敏处理，即不告知该属性在现实世界中的意义。

提交的文件里的预测值必须是 0（不会有交易）和 1（会有交易），文件格式如下：

```
ID_code,target
test_0,0
test_1,1
test_2,0
etc.
```

这次比赛参加队伍数高达 9038，反作弊检查之后剩余 8802，总人数超万人，可以说是 Kaggle 历史上参赛人数最多的比赛。

2. 数据清洗和模型训练

数据清洗， 是整个数据分析过程中不可缺少的一个环节，其结果质量直接关系到模型效果和最终结论。

---

[1] https://www.kaggle.com/c/santander-customer-transaction-prediction/overview
[2] http://en.wikipedia.org/wiki/Receiver_operating_characteristic

2.1 缺失值

缺失值是最常见的数据问题，首先对训练集和测试集进行检查，看其有无缺失值，代码如下所示，经检查训练集和测试集**均无缺失值**。

```
def missing_data(data):
    total = data.isnull().sum()
    percent = (data.isnull().sum()/data.isnull().count()*100)
    tt = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    types = []
    for col in data.columns:
        dtype = str(data[col].dtype)
        types.append(dtype)
    tt['Types'] = types
    return(np.transpose(tt))
missing_data(train_df)
# missing_data(test_df)
```

2.2 样本不均衡

查看一下训练集中“target”属性的 0 值和 1 值两类的分布情况，如下图 1 所示，值为 1 的样本有 179902 条，值为 0 的样本有 20098 条（占比 10.049%）。从图 1 中看出，这是个**样本不均衡**的问题。

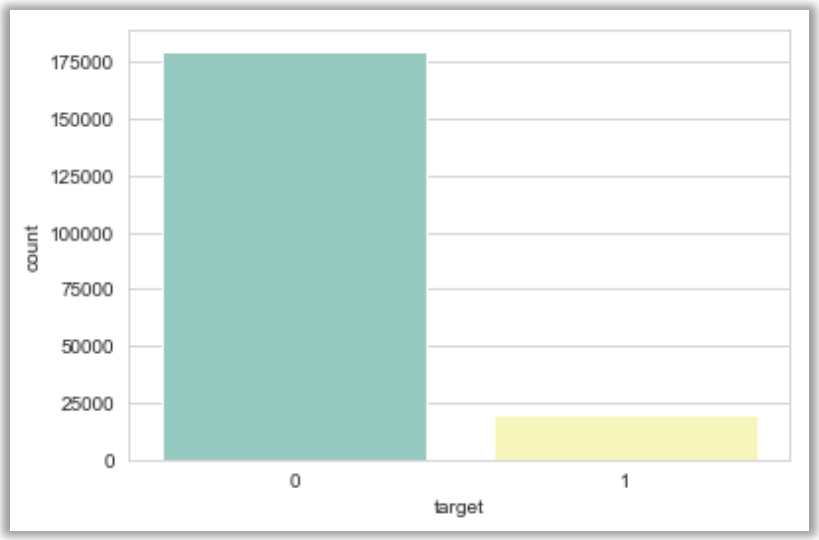

图 1　两类样本数目不均衡

2.3 线性判别分析模型 —— LDA

由于是个分类问题，并且特征全部是数值型，直接上线性判别分析模型。从图 2 中可以看出，将预测结果提交到 Kaggle 之后得分为 0.86114，排在第 6390 位。

这个排名还是相当靠后的，觉得不够满意，于是继续对特征进行分析。

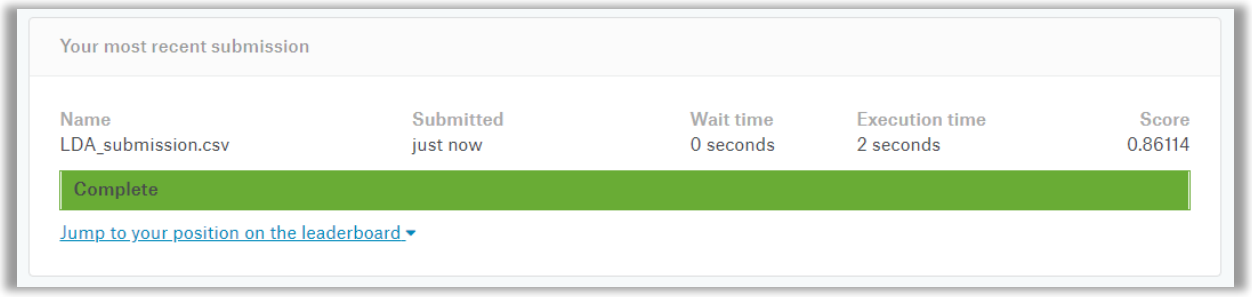

图 2　LDA 模型得分情况

2.4 特征相关性

计算训练集中特征之间的相关性（correlations）。特征相关性最大的是“var_183”和“var_189”，值仅为 0.009359。最低的是“var_75”和“var_191”，值为 2.703975e-08。所以这些特征之间的相关性很小。 如图 3 所示。

```
correlations.loc[(correlations['level_0']=='var_75') & ('var_191'== correlations['level_1'])]
```

	level_0	level_1	0
0	var_75	var_191	2.703975e-08

```
correlations.loc[(correlations['level_0']=='var_183') & ('var_189'== correlations['level_1'])]
```

	level_0	level_1	0
39790	var_183	var_189	0.009359

图 3　特征之间相关性最大值和最小值

2.5 检查训练集和测试集数据分布

对训练集和测试集的 200 个特征的统计信息进行分析。如下表所示展示了训练集和测试集的“var_0”和“var_1”两个特征的统计信息，从表中可以看出，训练集和测试集中最小值、最大值、均值以及方差都很接近，特征的**分布基本一致**。其余的 198 个特征也是如此，分布基本一致。

从这些统计数据上还可以观察到，不同特征之间的均值差距很大。

	训练集		测试集	
	var_0	**var_1**	**var_0**	**var_1**
count	200000	200000	200000	200000
mean	10.679914	-1.627622	10.658737	-1.624244
std	3.040051	4.050044	3.036716	4.040509
min	0.408400	-15.043400	0.188700	-15.043400
25%	8.453850	-4.740025	8.442975	-4.700125
50%	10.524750	-1.608050	10.513800	-1.590500
75%	12.758200	1.358625	12.739600	1.343400
max	20.315000	10.376800	22.323400	9.385100

利用 Python 的绘图库 seaborn 中的函数 distplot() 给 200 个特征绘制密度估计图。seaborn.displot()集合了 matplotlib 的 hist()与核函数估计 kdeplot 的功能，增加了 rugplot 分布观测条显示与利用 scipy 库 fit 拟合参数分布的新颖用途。

如下图 4 所示，展示的是训练集和测试集“var_0”和“var_1”两个特征的 KDE 图，从图中可以看出训练集和测试集的样本分布基本一致。

如下图 5 所示，展示的是训练集中目标属性“target”值为 0 的样本和值为 1 的样本“var_0”和“var_1”两个特征的 KDE 图。

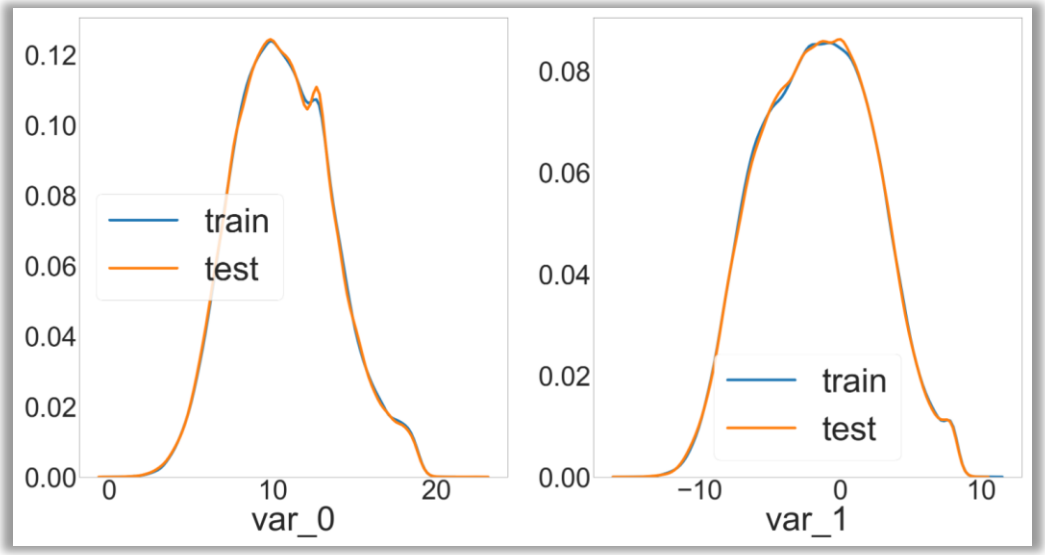

图 4　训练集和测试集样本密度图

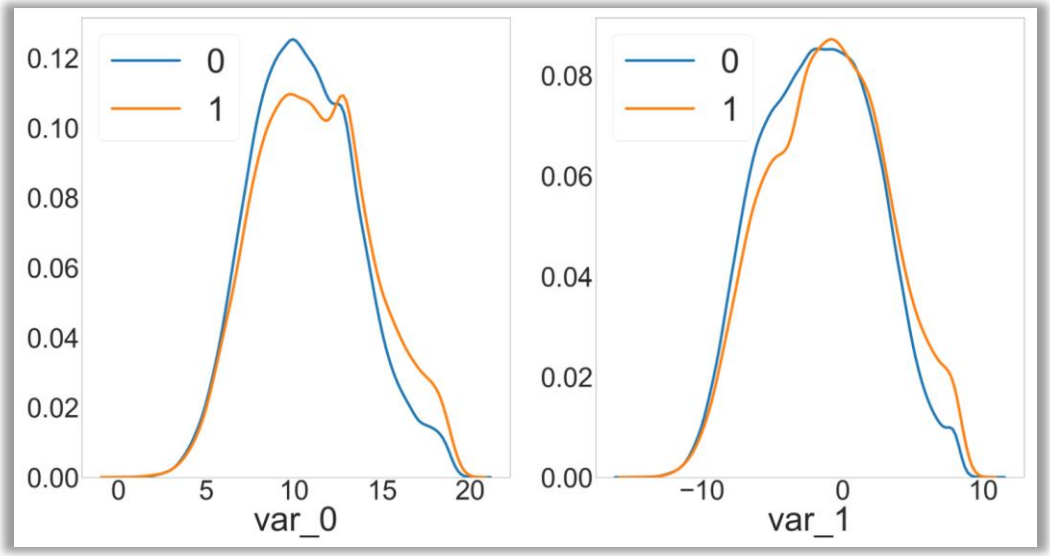

图 5　训练集不同类样本密度图

2.6 构造新特征

添加“sum”、“min”、“max”、“mean”、“std”、“skew”、“kurt”、“med”这八个新特征，分别代表样本 200 个原始特征的和、最小值、最大值、均值、方差、偏度（统计数据分布偏斜方向和程度的度量）、峰度（反映峰部的尖度）、中位数这八个统计度量。

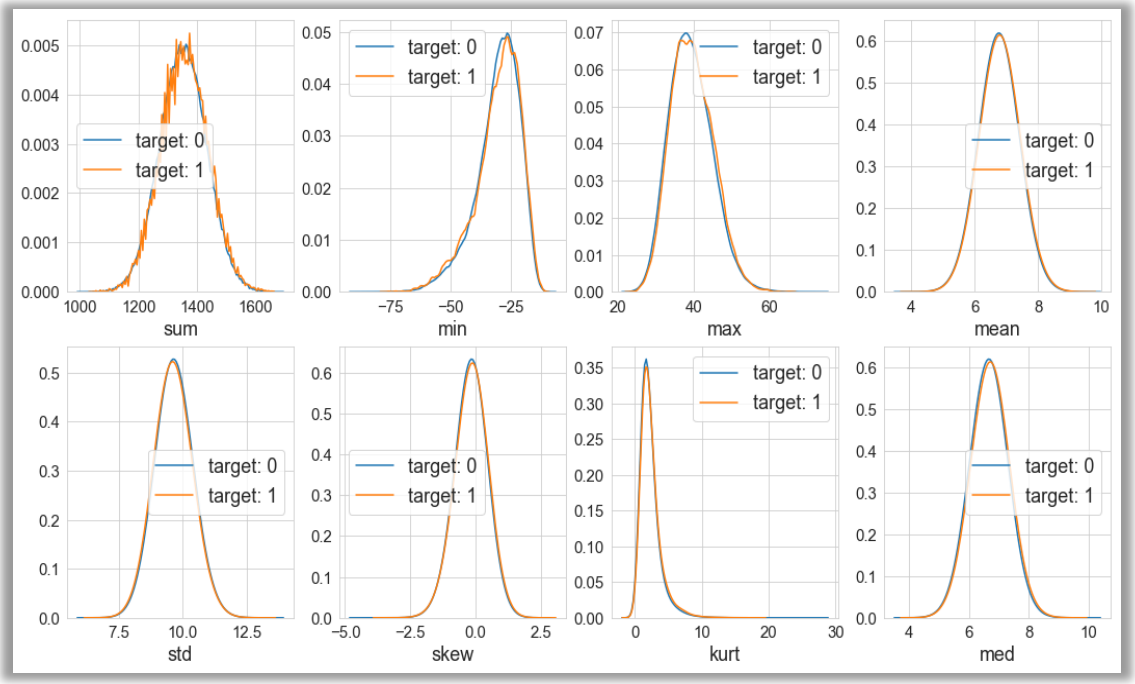

图 6　训练集两个类样本的八个新特征的密度分布图

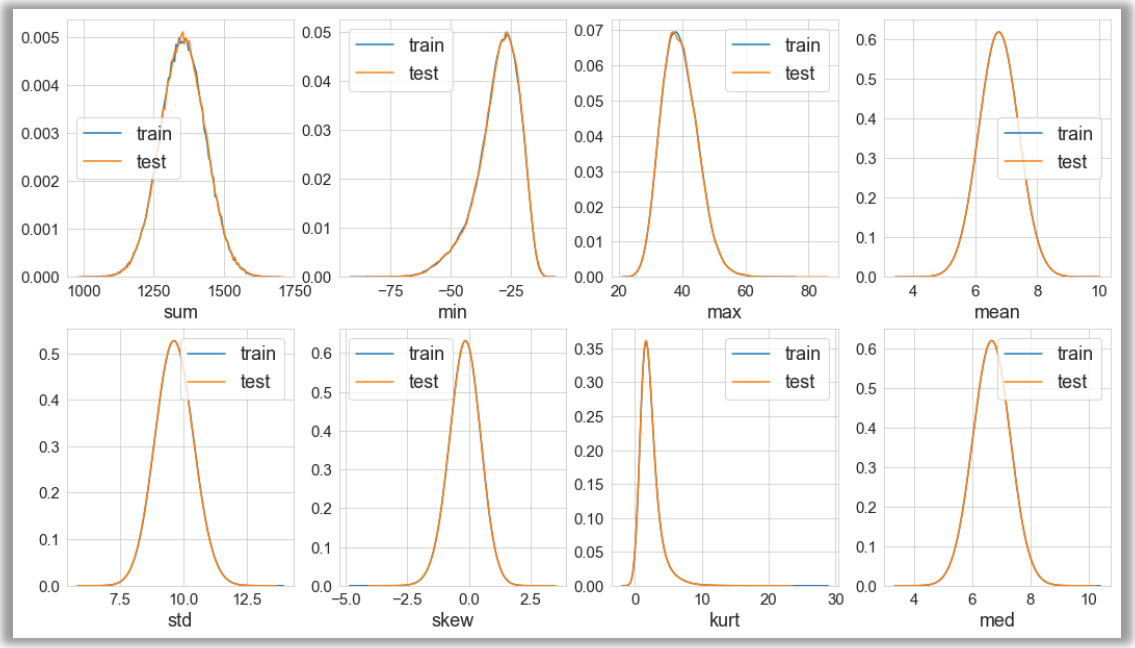

图 7　训练集和测试集的八个新特征的密度分布图

图 6 所示的是训练集“target”属性值为 0 和 1 的两个类的样本的八个新特征的密度分布图。图 7 所示的是训练集和测试集样本的八个新特征的密度分布图。

2.7 LightGBM —— 梯度 Boosting 框架

2017 年 1 月微软在 GitHub 上开源了 LightGBM[3]框架，发布之后三天内获得 1000+的 Stars，被评价为“速度惊人”、“非常有启发”、“支持分布式”、“代码清晰易懂”、“占用内存小”等，而且发布了官方文档[4]。

定义模型超参数，使用 LightGBM 来做预测。下表即为训练时所设置的参数。

```
param = {
    'bagging_freq': 5,
    'bagging_fraction': 0.4,
    'boost_from_average':'false',
    'boost': 'gbdt',
    'feature_fraction': 0.05,
    'learning_rate': 0.01,
    'max_depth': -1,
    'metric':'auc',
    'min_data_in_leaf': 80,
    'min_sum_hessian_in_leaf': 10.0,
    'num_leaves': 13,
    'num_threads': 8,
    'tree_learner': 'serial',
    'objective': 'binary',
    'verbosity': 1
}
```

在训练过程中，使用十折交叉验证来提升模型泛化能力。如下图 8 所示，展示的是训练完成后的最重要的三个特征，红色柱状图的长短代表了特征的重要程度。

图 8　模型中最重要的三个特征

从图 9 可以看到，将预测结果提交到 Kaggle 之后得分为 0.90021，排在第 3761 位。

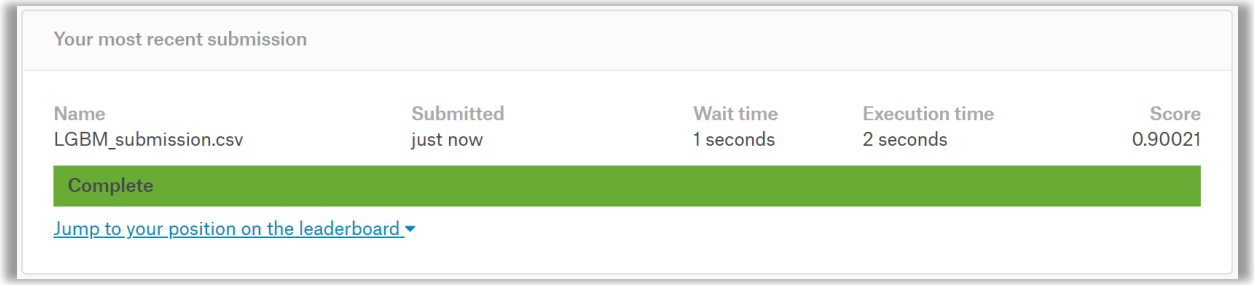

图 9　LightGBM 模型得分情况

[3] https://github.com/microsoft/LightGBM
[4] https://lightgbm.readthedocs.io/en/latest/

3. 深入挖掘数据 —— Fake Data

首先查看每个特征中的重复数值有多少。排最高的八个特征如下所示：

【训练集】

Feature	var_68	var_108	var_126	var_12	var_91	var_103	var_148	var_71
Max duplicates	1084	313	305	203	66	61	59	54
Value	5.0214	14.1999	11.5356	13.5545	6.9785	1.6662	4.0456	0.7031

【测试集】

Feature	var_68	var_126	var_108	var_12	var_91	var_103	var_148	var_161
Max duplicates	1104	307	302	188	86	78	74	69
Value	5.0197	11.5357	14.1999	13.5546	6.9939	1.4659	4.0004	5.7114

训练集和测试集中的相同列有着相同或非常相似的数量的最大重复属性值。事出反常必有妖，这一特性或许可以帮助我们来做一些预测。

以特征“var_0”为例，统计该特征不同值所出现的次数，之后构建一个新特征“var_0_count”，并且该特征属性值为对应“var_0”特征的属性值出现的次数。

借助于 seaborn.displot()对 Count 值为 1、2 和 3 的样本绘制特征“var_0”的概率密度图，如下图 10 所示。

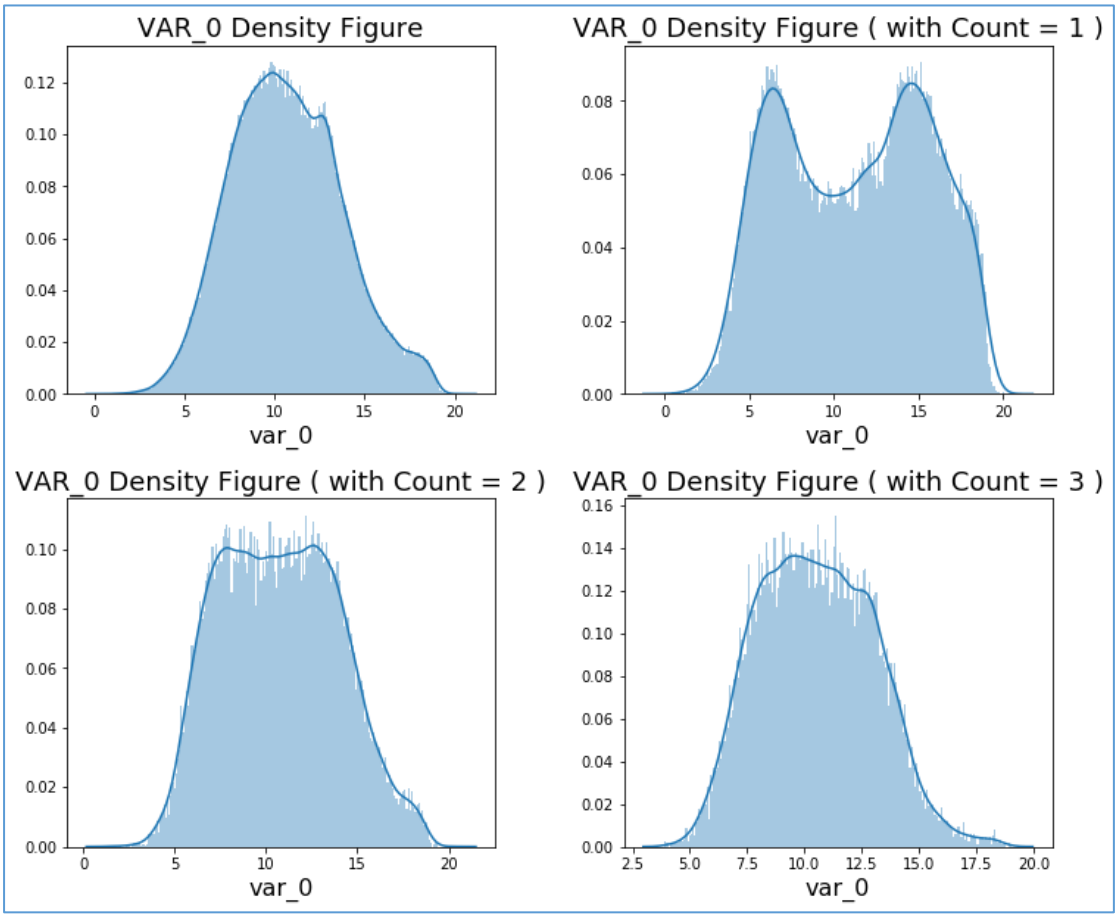

图 10　不同 Count 值对特征“var_0”分布的影响

从图 10 中可以明显看出，特征“var_0_count”的属性值为 1 的样本特征密度分布呈现出**较为明显的双峰分布！**说明特征“var_0_count”的属性值为 1 的这些样本比较特殊，下边重点研究下这 200 个特征中的每个特征的所有属性值出现次数为 1 的这些样本。

以特征“var_0”为例，利用 LightGBM 训练这一个特征，绘制热力图可得到如图 11 中的左图。如图 11 中的右图是两类样本的直方图。此时 AUC 值为 0. 53248。

由于前边提到特征中有很多重复数据，这里新建一个特征“var_0_count”将属性值替换为该特征的属性值出现的次数。利用 LightGBM 训练这两个特征，绘制热力图可得到如图 12 中的左图。如图 12 中的右图是两类样本的直方图。此时 AUC 值为 0.54785。

对比图 11 和图 12，明显看出 Count 属性对预测结果是有帮助的。

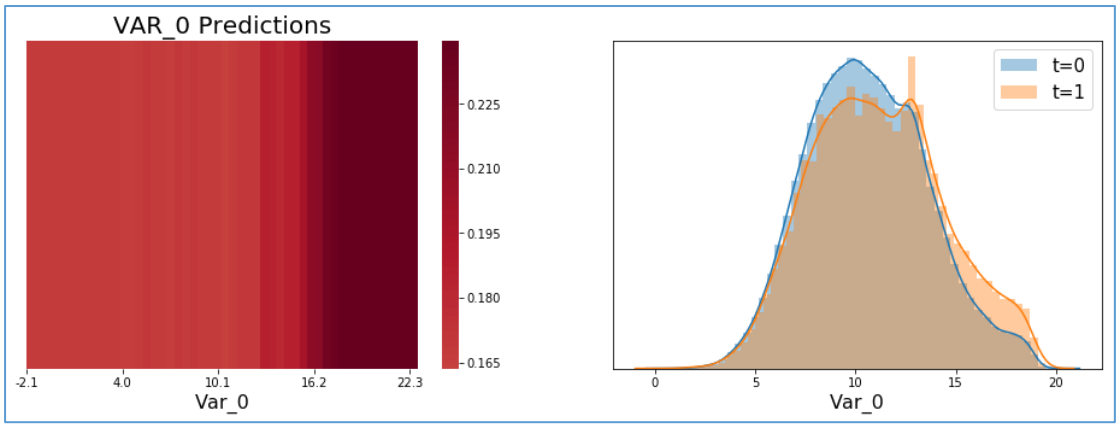

图 11　特征“var_0”的预测效果

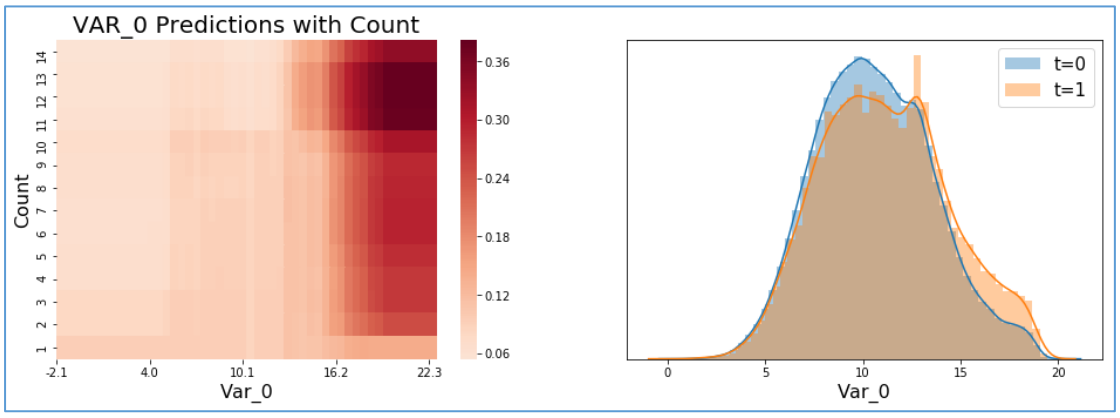

图 12　特征“var_0”和特征“count”的联合预测效果

这里采用一种集成策略。

首先对 200 个原始特征和它们的 Count 特征，两两组合，**使用 LightGBM 训练 200 个模型，并记录它们的预测值**。这里对于 200 个模型中的每个模型都采用五折交叉验证，将对应的验证集的预测值记录下来。测试集的预测值也会记录下来，这里将五折交叉验证中每一折的模型在测试集的预测值的平均值记录下来。

之后，**借助于 statsmodels 中的 Logit 模型进行融合**，以 200 个模型在训练集的预测值为训练数据，原始目标值仍旧作为目标值，进行训练。训练之后的模型，将 200 个模型对测试集的预测值作为测试数据，进行预测。

最后将结果提交到 Kaggle 平台上，如图 13 所示为得分情况。可以看到，Kaggle 平台给出了 0.92022 分，排名 72 名。

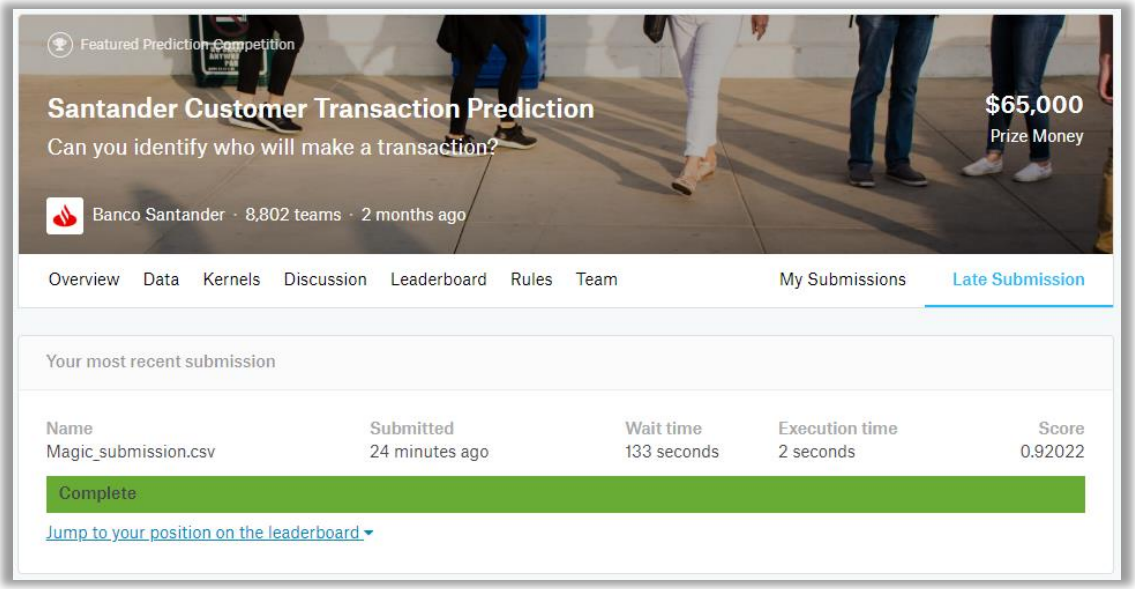

图 13　利用 Count 特征的集成模型得分情况

最终，在有着 8802 支有效队伍的比赛中，取得了 72 名的成绩，排名位列前 1%

4. 总结和结论

从这次比赛可以看出，数据集中的 fake data 在数据背后影响了模型的性能上限。

经历这次比赛，我真正体验到了数据挖掘的难度。通过对数据集的分析，并通过一些可视化来帮助理解数据，得到数据的信息。在这个过程中，不仅让我温故了之前学到的知识（例如线性判别分析 LDA），同时也提升了我的可视化以及绘图方面的技术知识。

另外，在数据预处理方面，比如缺失值的检查等，在通过这次比赛之后，也让我体会到了数据预处理的重要性。如果不是对数据深入的处理分析，那也不可能发现 fake data。**单纯的只是训练模型效果并不一定好（例如前边两种模型的表现都不是很优秀），有时候干净的数据才能充分地发挥模型的优势。**

至于模型的使用上，大多使用了一些常见的库，很多软件包集成了现有的许多机器学习和深度学习算法，这次比赛让我对这些工具的使用更加熟练了。

总的来说，此次比赛收获颇丰。