2023年秋-机器学期第一组报告

——大型多人在线角色扮演游戏中玩家充值金额预测

组号：1　　组长：杨镕争　　组员：陈翼飞、侯汝垚、王攀宇、张钊珲

一、问题背景与面临的挑战

问题背景：

游戏中玩家充值金额直接关系到企业的盈利，通过玩家历史游戏行为及充值记录预测未来充值金额，从而指导企业对游戏的优化以及对玩家定向推广，是游戏企业智能化的重要一步。

面临的挑战：

依据真实游戏玩家的行为数据，利用机器学习及数据挖掘相关方法，将用户的游戏行为与充值行为建立联系，对用户未来的充值金额进行预测。根据提供的原数据来看，多数玩家在游戏中并不充值，说明这个问题是一个长尾问题，即只有少量的类别含有较多样本，大部分类别的样本数都很少，这种样本不平衡带来的根本影响是：模型会学习到训练集中样本比例的这种先验性信息，导致在实际预测时就会对多数类别有所侧重。

二、实现的核心思想

首先将复杂的玩家游戏行为数据进行处理，从中提取出有价值的特征，并将其进行划分出训练集、验证集、测试集，之后选择四种不同的模型（LGBM、线性回归、神经网络、Stacking堆叠法），用处理好的数据进行训练，最后用训练好的模型推测最后一天的用户充值金额。

三、实现思路

1. 数据处理:

数据源自某公司的MMORPG游戏（已脱敏），为真实玩家行为信息，数据集涵盖多个维度的信息，包括历史7天内用户货币使用、角色升级、挑战副本、完成任务、上下线及充值记录等信息，部分数据信息如下：

文件名	文件内容	字段
role_id.csv	角色信息表	role_id - (玩家)角色id
		create_time - 角色创建时间
role_consume_op.csv	游戏货币使用表	level - 角色等级
		use_t1 - use_t4 - 四种游戏内货币的消耗情况，负数为获得，正数为消耗
		remain_t1 - remain_t4 - 消耗/获得三种游戏内货币 的现存值
		mtime / dt - 操作时的时间戳/日期
role_evolve_op.csv	角色道具等级升级表	type - 升级类型
		item_id - 使用了何种道具进行升级（类似强化石）
		num - 消耗了多少道具
		old_lv/new_lv - 升级前/后等级

图 1 历史七天内玩家游戏行为列表（部分）

我们从 csv 文件中提取出特征，包含了 id，创建时间，不同道具的使用情况，升级次数，升级级数，在线时间，任务类型，副本次数，离线原因，充值等情况，并将其转化成可以用来训练的形式。

在处理数据的过程中，我们发现第七天充值量中，除了大部分的 0（大部分玩家玩游戏是不会充值的），剩下的充值记录中，有 115 项 6 元以及 44 项 30 元，而其他的充值记录只有不到 30 条。这大概率说明游戏存在首充机制（6 元）与月卡机制（30 元）。然而根据经验，大部分选择这种少量充值的玩家只是为了一开始少量金额提供的福利，并不会继续充值，因此这些充值记录更像是噪声，会干扰最后判断的结果。

考虑到这些噪声的影响，我们采取了正则化处理，它有助于平滑模型，减少对训练数据中噪声的过度拟合。正则化都是在数据训练的过程中，对权重后面加上一个损失项，不断减少权重的值。正则化可以分为两种：L1 正则化与 L2 正则化。L1 正则化即 Lasso 正则化，它会在训练的过程中不断使一些特征的权重减小，并且可能变为 0；L2 正则化为 Ridge 正则化，相比于 L1 正则化要稍微缓和一些，尽管也会减小权重，但是不会将其减小到 0。

2. 数据训练：

- **LGBM(决策树模型)：**

LightGBM 是一款基于梯度提升决策树的分类器,支持多种参数和格式。在实现时采用 Leaf-wise 生长策略——选择当前最大梯度的叶子节点进行生长，这有助于提高模型的深度，更好地捕捉数据的复杂性。同时使用了直方图算法：通过对连续特征进行分桶，减少了计算复杂度，在大规模数据

集上的性能更为出色。

以下为使用该分类器时用到的部分重要参数：

num_leaves: 每棵树的最大叶子节点数

max_depth: 每棵树的最大深度

min_child_samples: 每个叶子节点最少样本数

min_child_weight: 每个叶子节点最小的样本权重和

subsample: 每次迭代使用的训练样本的比例

learning_rate: 学习率

feature_fraction: 每次迭代使用的特征的比例

reg_alpha

通过设置合理的参数值，可以让训练的效果达到更好，下图为本组实验最终选择的参数数值：

```
params = {
    'objective': 'regression',
    'metric': {'rmse'},
    'boosting_type' : 'gbdt',
    'learning_rate': 0.05,
    'max_depth' : 5,
    'num_leaves' : 8,
    'feature_fraction' : 1,
    'subsample' : 0.8,
    'seed' : 114,
    'num_iterations' : 3000,
    'nthread' : -1,
    'verbose' : -1,
    'reg_alpha': 0.1,
    'reg_lambda': 0.2
}
```

图 2　LGBM 分类器的参数取值

下面详细地解释 LGBM 的几个基本原理：首先用直方图算法将特征离散化，即将特征值区间分成若干个小区间，将每个区间的值赋成该区间内的样本数，这可以使得算法的内存消耗更少、计算代价更小。接着采用 Leaf-wise 算法，每次从已生成决策树的所有叶子节点中找到分裂增益最大的一片叶子，然后分裂，如此循环。该过程还设定了最大深度，防止生长出较深的决策树导致的过拟合。在处理样本的时候，LGBM 采用单边梯度采样，通过策略地删除大部分小梯度的样本，在减少数据量和保持算法精度上维持了平衡。

- **线性回归：**

线性回归大体上采用了两种思路，一种是基于 forward 和 backward 的梯度下降法，设置学习率 0.1，迭代 1000 次，得到最终的参数 w。

```
class Linear_Model:
    def __init__(self, learning_rate=0.1, num_iterations=1000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None
```

图 3　梯度下降法的参数设置

另一种则是基于最小二乘法，基本思路就是对损失值求关于参数集 w 的偏导，再取偏导数为 0 时的参数集 w，用它来推断最后一天的充值金额。多元线性回归的损失值为“预测值减去真实值的平方”，如下公式:

$$\mathrm{L}(\omega) = (\mathrm{Xw} - \mathrm{Y})^{\mathrm{T}}(\mathrm{Xw} - \mathrm{Y}) = ||\mathrm{Xw} - \mathrm{Y}||_2^2$$

对损失值求偏导，得到：

$$\frac{\partial}{\partial w} L(\omega) = 2X^T(Xw - Y)$$

在偏导为 0 时，可以算出所求的参数集w^*为：

$$w^* = (X^T X)^{-1} X^T y$$

在最小二乘法实现的基础上，我们在其训练过程中还加入了正则化处理，通过减少噪声的影响以求获得更好的训练效果。

```
class RidgeRegression:
    def __init__(self, alpha=1.0):
        self.alpha = alpha  # 正则化参数
        self.weights = None
```

图 4　Ridge 正则化处理参数设置

```python
class LassoRegression:
    def __init__(self, alpha=1.0, max_iter=1000, tol=1e-4):
        self.alpha = alpha  # 正则化参数
        self.max_iter = max_iter  # 最大迭代次数
        self.tol = tol  # 收敛阈值
        self.weights = None
```

图 5　Lasso 正则化处理参数设置

- 神经网络：

采用简单的三层架构，神经元数量分别是 32、16、1，分别用 Relu、Relu 与线性激活函数来激活。尽管简单，但也尝试过其他更复杂的模型，效果均不如该模型。

```python
# 创建模型
model = Sequential()
model.add(Dense(32, input_dim=num_features, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='linear'))

# 编译模型
optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(optimizer='adam', loss='mean_absolute_error')
```

图 6　神经网络的搭建

- Stacking 堆叠法：

该算法模型效果较好、可解释性强、适用于复杂数据，属于融合方法。具体思想即：先将数据同时发给上述的三个模型（LGBM、线性回归、神经网络），我们不妨称这三个算法为 level 0 层，训练完毕后，将每个模型的输出拼凑成新的特征矩阵，在输出给一个新的模型 X（level 1），这个模型 X 的输出就是最终的结果。Level 0 上的算法们的职责是找出原始数据与标签的关系、即建立原始数据与标签之间的假设，需要强大的学习能力；而 level 1 上的算法的职责是融合各个模型做出的假设、并最终输出融合模型的结果，相当于在寻找“最佳融合规则”。

在 level 1 层，我们采用了最小二乘法来训练不同模型得到的结果，获取最终成绩。

3. 分数换算：

最终得分获得基于预测值与真实值的误差。首先计算出预测结果与实际结果的均方对数误差 msle。均方对数误差是回归算法的一种常用评估指标：

$$\mathrm{MSLE}(y,\hat{y}) = \frac{1}{N_{sample}} \sum_{i=1}^{N_{sample}} (\ln(1+y_i) - \ln(1+\hat{y}_i))^2$$

再使用归一化函数，得到最终的分数：

$$\mathrm{Point} = \frac{1}{1+\mathrm{msle}}$$

四、实验结果对比及分析

1. LGBM：

- 用 pay_sum_day 或 pay_log 作为特征，得分：0.7221207952368351；
- 将 pay_sum_day 与 pay_log 都作为特征，得分：0.7275144407799953；
- 在前面的基础上引入正则化，得分：0.7283204884782382。

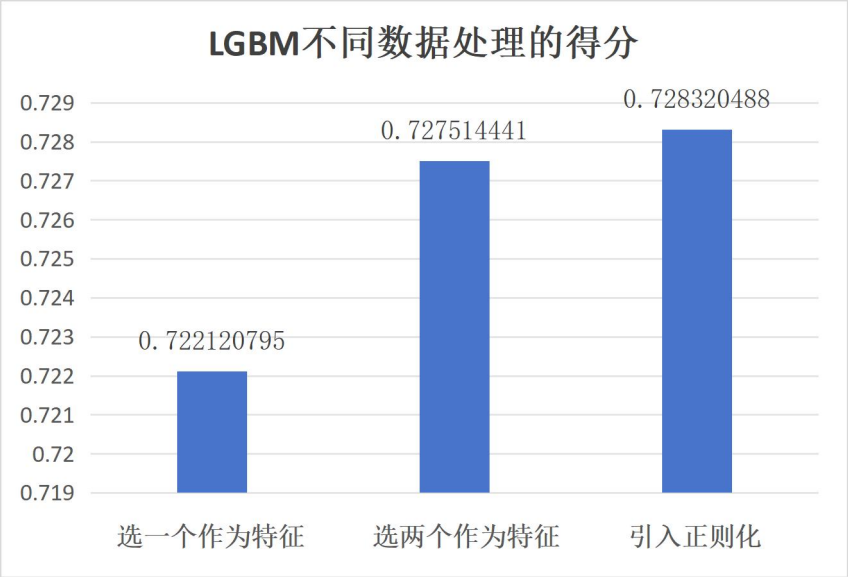


图 7 LGBM 的得分情况

2. 线性回归：

- 仅用简单的线性回归，得分为 0.598338631716166
- 使用最小二乘法，得分为 0.6531393057132168
- 使用 Ridge 正则化后，得分为 0.6531401573790739
- 使用 Lasso 正则化后，得分为 0.656765813131181

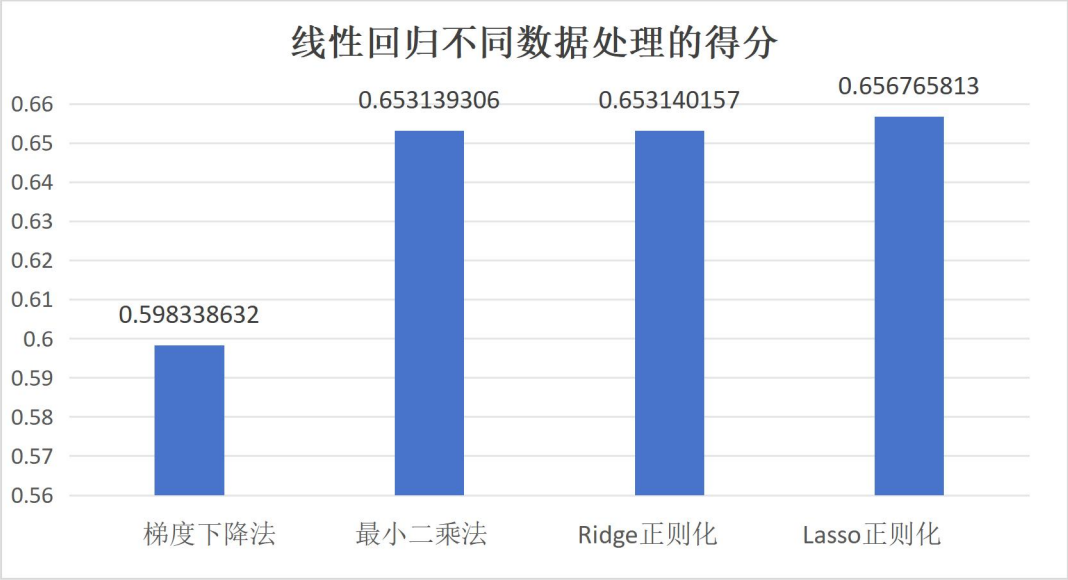


图 8 线性回归的得分情况

最小二乘法直接求解最小二乘法的闭式解，即通过对损失函数的偏导数等于零，得到模型参数的解析解。这种方法在特征数不大时效果较好，但对于大规模数据集可能计算复杂度较高。

梯度下降通过迭代更新参数，每次迭代都根据损失函数关于参数的梯度进行参数调整。梯度下降适用于大规模数据集，因为每次迭代只需要使用一部分数据（小批量梯度下降）或一个样本（随机梯度下降）。这也造成了最小二乘法对异常值比较敏感，因为它会受到离群点的影响，而梯度下降相对较为鲁棒，通过调整学习率等参数，可以减缓异常值的影响。因此，在数据规模较小，特征数不多的情况下，可以考虑使用最小二乘法，尤其是在有解析解的情况下。在数据规模较大，特征数较多的情况下，梯度下降是一种更为常用的优化方法，尤其是在深度学习等领域。

3. 神经网络：

这里用了非常简单的三层结构，神经元数量为 32,16，1，激活函数分别是 Relu、relu，线性激活，最后使用 Adam 优化器，均方差作为损失函数进行编译，得到的效果却比尝试的其他更复杂的神经网络效果要好。但在训练过程中发现两个很严重的问题：一是特征数量太多导致不收敛，loss 的数量级爆炸甚至为 NaN，在修改部分

- 将全部的特征用来训练，最终得分为 0.6176425659671799
- 筛选特征之后再训练，最终得分为 0.6780988264033687

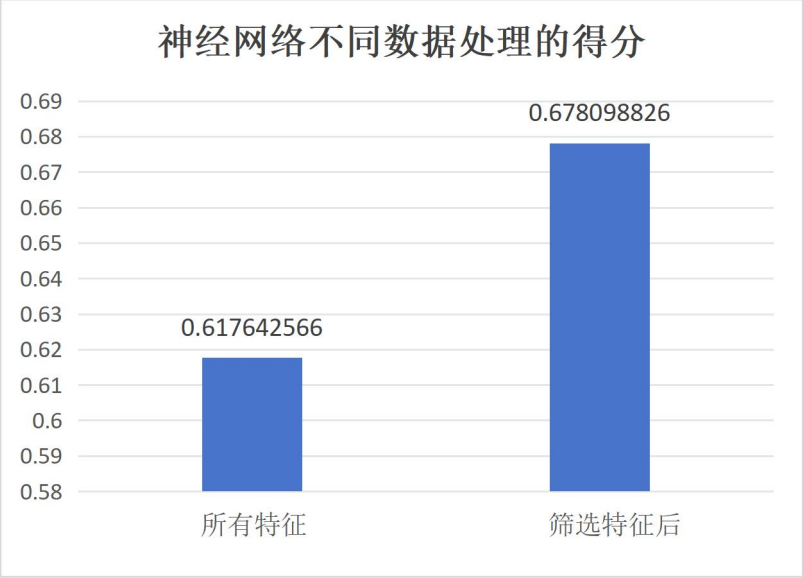
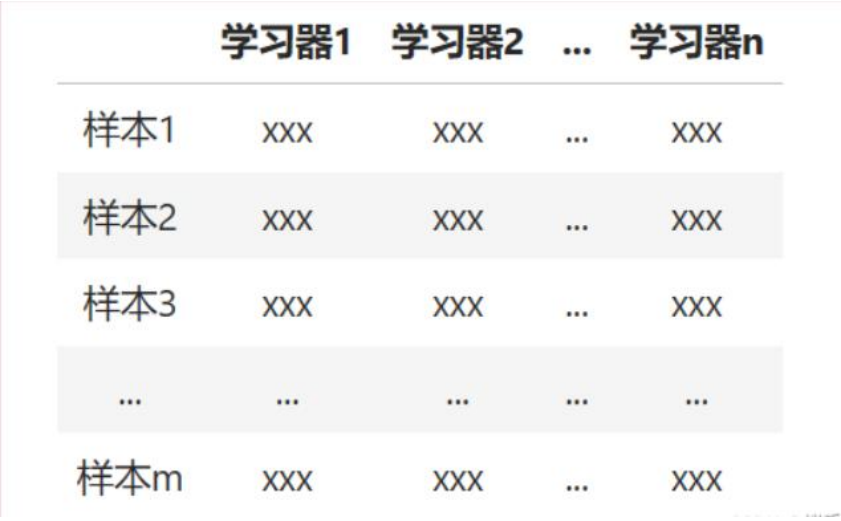


图 9 神经网络的得分情况

第二个问题是神经网络的误差非常大，同样的代码可能有 0.04 的误差，波动太大。在调查资料后，引入了早停法，监控验证集上的性能，可以在模型性能不再提升时停止训练，从而避免过拟合。这种方法把误差降低到 0.01。此外也尝试了 Batch Normalization 批标准化，在每一层后加上一层标准化，确实起到了增加鲁棒性的效果，但得分甚至掉到了 0.60，因此弃用。

4. Stacking 堆叠法：

集成模型的基础模型选择了线性模型和神经网络模型，元模型选择了线性模型。



	学习器1	学习器2	...	学习器n
样本1	xxx	xxx	...	xxx
样本2	xxx	xxx	...	xxx
样本3	xxx	xxx	...	xxx
...	...	...	...	...
样本m	xxx	xxx	...	xxx

图 10 集成学习的原理示意图

对于同一个样本，先经过两个模型分别预测，得到的两个结果再作为元模型的输入进行训练，得到输出，这就是堆叠法的原理。此外还有投票法、求均值法等不需要元模型的方法，但考虑到鲁棒性交叉，选择了堆叠法。

集成方法最终得分为 0.6842220104909934，由于仅有一个数据，不再单独画图。该法仅次于 LGBM 模型，比线性回归与神经网络模型的得分要更好。刚才提到的神经网络鲁棒性差的问题在这里也得到解决，集成模型的误差范围在 0.001 数量级。

5. 对比分析：

我们将每种模型的分数放在一张图表中：

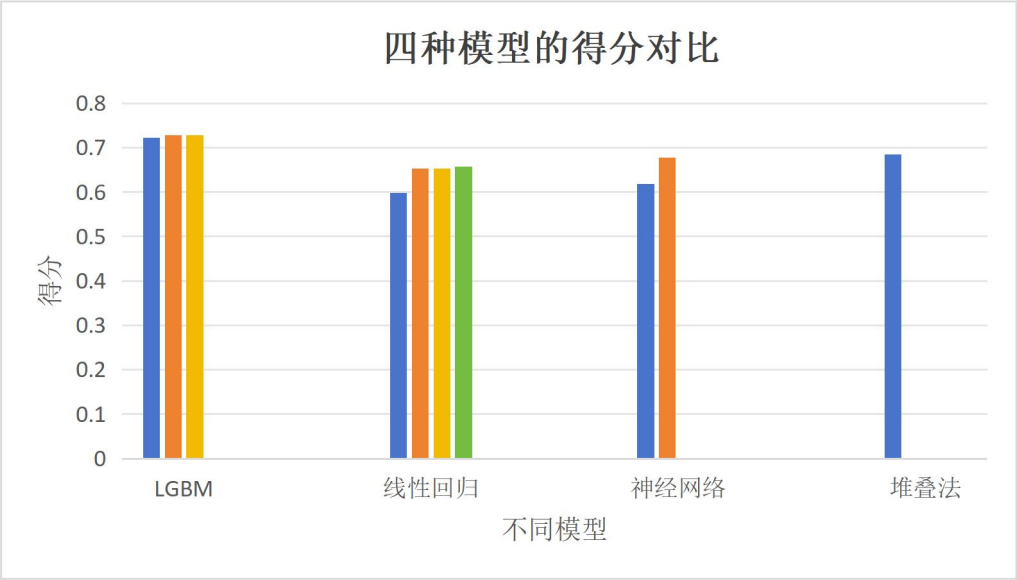


图 11 四种模型的得分情况比较

不论是从最高分、最低分来看，线性回归是三种模型中得分最低的模型，而 LGBM 则是三种模型中得分最高的，而且 LGBM 的任意一种数据处理方法得分都显著高于另外两个模型，说明此类问题适合用 LGBM 分类器来预测。

另外，前三种方法中可能采用了不同的数据处理方法，这些方法对分数的改善并不多，可能只有千分之几的提高，但是这个提高不能忽视，甚至说是意义重大的。因为通过下面两次排名与分数的对比，我们可以发现尽管分差只有 0.002 左右，但是排名却相差特别多。

5	-	剪秋本官的头好痛	1	2023-12-08 18:15	0.72157934
6	-	鸡泣学习	1	2023-12-08 17:18	0.72142410
7	-	我在USTC不玩原神	1	2023-12-08 15:24	0.72107261
23	↓ 8	疯狂大马哈	1	2023-12-09 22:47	0.71978776
24	-	鸡泣学习	1	2023-12-09 23:58	0.71913571
25	-	default13269584	1	2023-12-09 20:30	0.71876473

图 12 排名与分数关系

最后，还有一个显著的感受，就像老师上课说的一样，机器学习是一个务实的学科。就比如在神经网络模型中，采用复杂神经网络未必效果更好，反而简单的神经网络得到结果分数更高；另外，线性回归模型采取了合理的数据处理方式，其分数也有可能比神经网络的效果好，即复杂的模型未必会有更好的表现。因此，在机器学习的道路上，应当追求“务实”而非复杂度。

五、主要参考文献

- 2023 ZLab 学习赛/大型多人在线角色扮演游戏中玩家充值金额预测-baseline.ipynb
- 一篇文章完全搞懂正则化（Regularization）.CSDN 博客.取自：
  https://blog.csdn.net/weixin_41960890/article/details/104891561
- 线性回归模型详解(Linear Regression) . CSDN 博客. 取自：
  https://blog.csdn.net/iqdutao/article/details/109402570
- 机器学习—LightGBM 的原理、优化以及优缺点. CSDN 博客. 取自
  https://blog.csdn.net/weixin_46649052/article/details/119604545

六、小组成员、学号

杨镕争 2021k8009929022

陈翼飞 2021k8009925007

侯汝垚 2021k8009929034

王攀宇 2021K8009929018

张钊珲 2021K8009929029

七、小组成员分工及贡献

杨镕争：题目选定、初始数据处理、线性模型的设计与实现。

陈翼飞：初始数据处理、撰写实验报告、集成方法的设计与实现。

侯汝垚：线性模型的设计与实现、决策树模型的设计与实现。

王攀宇：决策树模型的设计与实现，神经网络模型的设计与实现。

张钊珲：神经网络模型的设计与实现、制作汇报 PPT。