“用户人口属性预测”算法实现说明文档

1. 赛题理解

(1) 待解决的问题

对于手机设备厂商，获取当前手机用户的人口属性信息（demographics）非常困难，当前华为手机 3.5 亿用户中，大概只有 5000 万用户的性别和年龄信息，如何基于用户的手机及使用偏好准确地预测其人口属性信息是提升个性化体验、构建精准用户画像的基础。手机用户的人口属性（如性别、年龄、常驻地等）数据一方面可以被用于个性化推荐服务，提升用户体验，另一方面可以用于手机用户群画像分析，帮助厂商了解产品的人群定位，优化产品设计。

年龄是用户人口属性的重要维度，本次比赛任务为根据用户的手机使用行为习惯来预估用户所处的年龄段。每个用户（以唯一 ID 标识）对应唯一的年龄段。年龄段有 6 种划分，分别代表不同年龄段，分别为：小于等于 18 岁， 19-23 岁， 24-34 岁， 35-44 岁， 45-54 岁，大于等于 55 岁。参赛者根据华为提供数据构建预测模型进行年龄段预估，在测试数据集上给出预估结果。

(2) 思路

本赛题是一个典型的多分类问题，数据包含多个数值型、类别型以及多值离散型特征，对于刚刚接触此类数据量较大且复杂问题的同学来说，赛题不仅对硬件性能有较高的要求，更对团队协作和业务能力发出一个不小的挑战。在数据的处理上，对于多值离散特征，可以使用词频矩阵、词向量或 TF-IDF 矩阵进行处理,特征维度过高时可以使用 PCA 或 LDA 进行降维。

多分类预测方案的对比：

- 在预测中可以将多分类问题看做多个二分类问题（One-Vs-All），选择其中一个类别为正类（Positive），使其他所有类别为负类（Negative）。比如第一步，我们可以将三角形所代表的实例全部视为正类，其他实例全部视为负类。在预测阶段，每个分类器可以根据测试样本，得到当前正类的概率。即 $P(y = i \mid x; \theta)$，i = 1, 2, 3..6。选择计算结果最高的分类器，其正类就可以作为预测结果。
- 相比于 One-Vs-All 由于样本数量可能的偏向性带来的不稳定性，One-Vs-One 是一种相对稳健的扩展方法。对于 6 分类问题，我们像举行车轮作战一样让不同类别的数据两两组合训练分类器，可以得到 6 个二元分类器。任何一个测试样本都可以通过分类器的投票选举出预测结果，这就是 One-Vs-One 的运行方式。
- Softmax：我们用 Sigmoid 函数将一个多维数据（一个样本）映射到一个 0 - 1 之间的数值上，那有没有什么方法从数学上让一个样本映射到多个 0 - 1 之间的数值呢？有！我们可以通过 Softmax 函数，使所有概率之和为 1，是对概率分布进行归一化。在处理一些样本可能从属多个类别的分类问题是，使用 one vs one 或 one vs all 有可能达到更好的效果。Softmax 回归适合处理一个样本尽可能属于一种类别的多分类问题。

(3) 难点

- 多值离散型特征的处理
- 基础特征的挖掘

- User_app_usage 样本缺失的处理
- 模型的搭建，融合方案的制定

2. 探索性数据分析

（1）数据缺失分析与处理

经过 EDA 的统计分析，我们发现整个数据情况缺失还是比较少的。主要的缺失分布在用户基础属性表 user_basic_info 中。字体，RAM、ROM 剩余和 ct 的缺失较为明显,具体如下：

- user_behavior_info：无缺失；
- user_app_actived 和 app_info：无缺失；
- user_basic_info：缺失情况如下图 1，使用**众数**进行填充处理。

	total_missing	percent
fontSize	615570	24.50%
ramLeftRation	253072	10.07%
romLeftRation	216444	8.61%
ct	141553	5.63%
ramCapacity	30946	1.23%
romCapacity	30946	1.23%
city	10519	0.42%
os	1228	0.05%

图 1 user_basic_info 缺失情况

- user_app_usage：**样本缺失严重**，训练+测试总样本数 2512500，user_app_usage 表中缺失 519833 个样本的 APP 使用情况，**缺失比例达 20%**. 我们有两个解决方案，
  - 方案一：将样本分为两个部分，无缺失样本使用全量特征预测+有缺失样本排除 app_usage 类特征进行预测。
  - 方案二：采用已有构造的特征预测缺失样本的 app_usage 数据，再跑全量样本训练

（2）标签分布

训练集和测试集情况，为了保证测试集能够与训练集同分布，故进行了统计：

表 1 训练样本标签分布

表	行数	列数	年龄分布
age_train	2010000	2	1 60000 2 400000 3 600000 4 500000 5 300000 6 150000
age_test	502500	1	待预测

经线上测试，我们发现：**测试集标签分布与训练集一致！**即需预测的数据分布如下图 2：

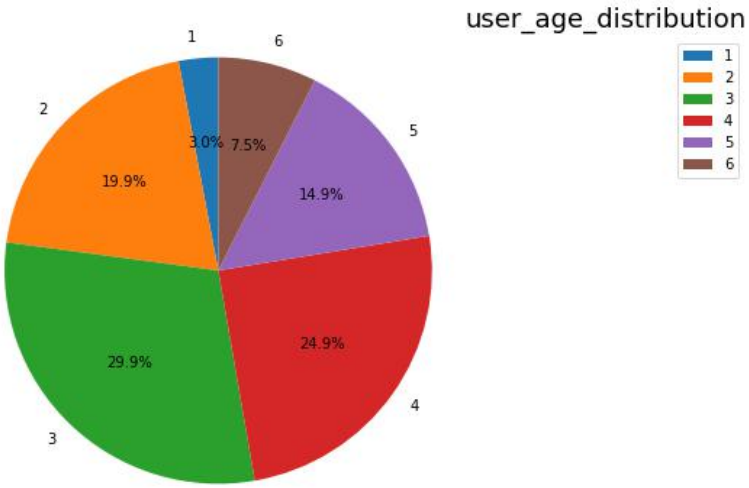


图 2　训练集-测试集标签分布一致

（3）数据预处理

在基础数据集的探索上，我们首先进行一部分数据类型和分布的探索，得到了一些结论和总结。

在数据类型探索方面，由于部分类别特征的类别之间存在大小关系，故也可以直接将其当做离散特征进行处理和直接输入到模型中进行训练。

另外，部分特征的数值类型与业务类型之间存在不一致，比如说用户行为表中功能使用次数中大部分值都为浮点型，而业务逻辑中次数应该为整型。故在前期的处理中可以选择性的将其类型进行规整，不过，后续发现，是否处理对于测试效果没有太大影响，故进行了其他的处理，后续会提到。

基础特征的预处理方法见下表 2，不同的算法模型可能对应不同处理方案，若处理是 category，则意味着使用的模型算法（若可以）直接指定类别特征，“+Rank”即保持原有数据的同时加入排序特征，“OrdinalEncoder”即将字符串型类别特征进行顺序编码，然后可以通过树模型直接指定类别特征或作为数值型直接参与训练（测试发现两种处理对树模型来说差别不大，可以作为模型融合方案分别尝试）。

表 2　基础特征预处理

表	字段	类型	数值类型	预处理
user_basic_info	性别（gender）	类别	int8	One-hot 或 category
	常住地（city）	类别	str	OrdinalEncoder
	手机型号（prodName）	类别	str	OrdinalEncoder
	手机 ram 容量（ramCapacity）	数值	float	+Rank
	ram 剩余容量占比（ramLeftRation）	数值	float	+Rank
	rom 容量（romCapacity）	数值	float	+Rank
	rom 剩余容量占比（romLeftRation）	数值	float	+Rank
	手机颜色（color）	类别	float	+Rank
	字体大小（fontSize）	类别	float	+Rank
	上网类型（ct）	类别	str	One-hot 或 OrdinalEncoder
	移动运营商（carrier）	类别	str	One-hot 或 OrdinalEncoder
	手机系统版本（os）	类别	float	+Rank

user_ behavior_info	开机次数（bootTimes）	数值	int	+Rank
	手机 A 特性使用次数（AFuncTimes）	数值	float	+Rank
	手机 B 特性使用次数（BFuncTimes）	数值	float	+Rank
	手机 C 特性使用次数（CFuncTimes）	数值	float	+Rank
	手机 D 特性使用次数（DFuncTimes）	数值	float	+Rank
	手机 E 特性使用次数（EFuncTimes）	数值	float	+Rank
	手机 F 特性使用次数（FFuncTimes）	数值	float	+Rank
	手机 G 特性使用情况（FFuncSum）	数值	float	+Rank
user_app_actived	应用标识（appId）	类别	str	W2V 或 TF-IDF
user_app_usage.	使用时长（duration）	数值	int	groupyby.agg()
	打开次数（times）	数值	int	groupyby.agg()
	使用日期（use_date）	时间	str	每天/周/工作日/周末
app_info	应用类型（category）	类别	str	清洗后 OrdinalEncoder

异常数据处理方法：

- **用户行为表**中各大功能的使用次数中存在负值，不符合业务场景实际意义，此处有 3 种方案：一是不处理，该部分数据有可能是错误记录导致，可以让模型自行识别；二是将该部分数据直接处理成众数 0 值，三是统一使用-1 填充，即将异常值作为独特一种取值。经测试发现，三种处理方案之间差别不大，意外的是不做任何处理的方案在准确率分数上反而有万分位上的优势，鉴于这种结果，我们此处选择不处理。
- 另外**用户行为表**中使用次数特征大部分都属于浮点型，若真实数据，则需要取整满足业务需求，而且数值分布差距不大，无太大意义，猜测原因可能两种：一是该数据情况是由于长期数据（比如一年）取一个月的平均值造成；二是数据做了等值缩小，出题方需要我们自行梳理。此处我们对于浮点型行为数据均进行了 10 倍放大操作，这样有利于我们进行 30 天取均值、取平方值等操作的便利性（小于 1 的浮点数取平方更小，这不符合行为次数类数据业务直觉）。
- 手机颜色数据中出现“备件颜色”值，不做处理，将其作为独特一种颜色。
- User_app_usage 中统计用户使用 APP 的时长和次数会出现少数偏离正常水平的数值，此处使用了添加排序特征的方式对数据做了平滑处理，因为排序特征对于模型的稳定性提升有比较好的效果。

（4）数据单变量分析

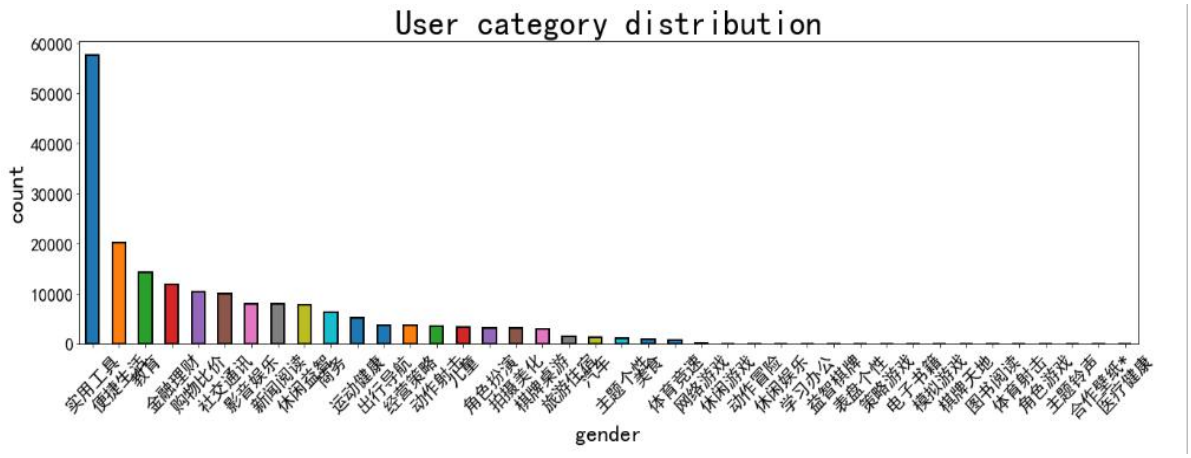


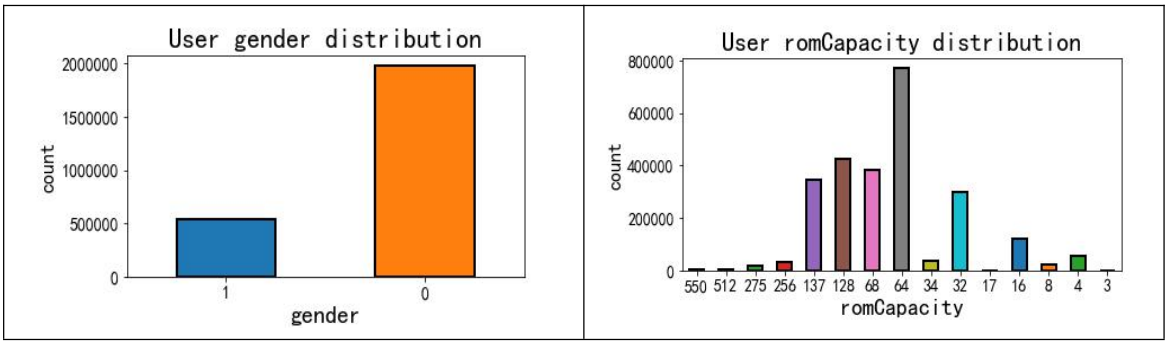
图 3　App 类型分布图

从数据分布的可视化情况可以看出，用户使用的 App 类型情况大致服从正态分布，只有少部分类型之间存在着表意重复的脏数据，经过数据清洗和数据规整，参考“华为手机应用市场”分类方法（如下图 4），我们将所有的 App 类型归纳为 23 类，从而将数据的长尾分布的影响降低。

图 4　华为应用市场分类方法

另外，在数据类别较少的下列 6 个特征中（如下图 5），我们可以发现：

- 数据中性别 0 的用户总量远大于性别 1 的情况，反应到用户年龄似乎不合常理，此处应该是有一个用户注册认证习惯导致，猜测用户 0 为男性（更多的使用相关认证服务）。
- RomCapacity 手机 ROM 容量中选用 64G 的用户最多，其他数据相应服从正态分布，用户逐渐减少，不过，可以看到 34G 和 17G 的用户存在着一些异常，该两种 ROM 配置比较少见，猜测可能是小众品牌或者数据存在标准误差。
- RAM 主要以 6G 和 4G 为主，数据分布与样本标签分布有些类似，后续可以通过多变量分析两者之间关系。
- 运营商的选择中大致与目前四家运营商的市场份额相同。
- 接入方式上，用户主要还是以 wifi 和 4G 为主，不过也存在极少数的 3G 以及 2G 网络用户。
- 系统版本上，用户主要使用 9.0,8.0,7.0 等整数版本号系统。这或许可以与机器型号之间做关联，得出手机型号的新旧情况。



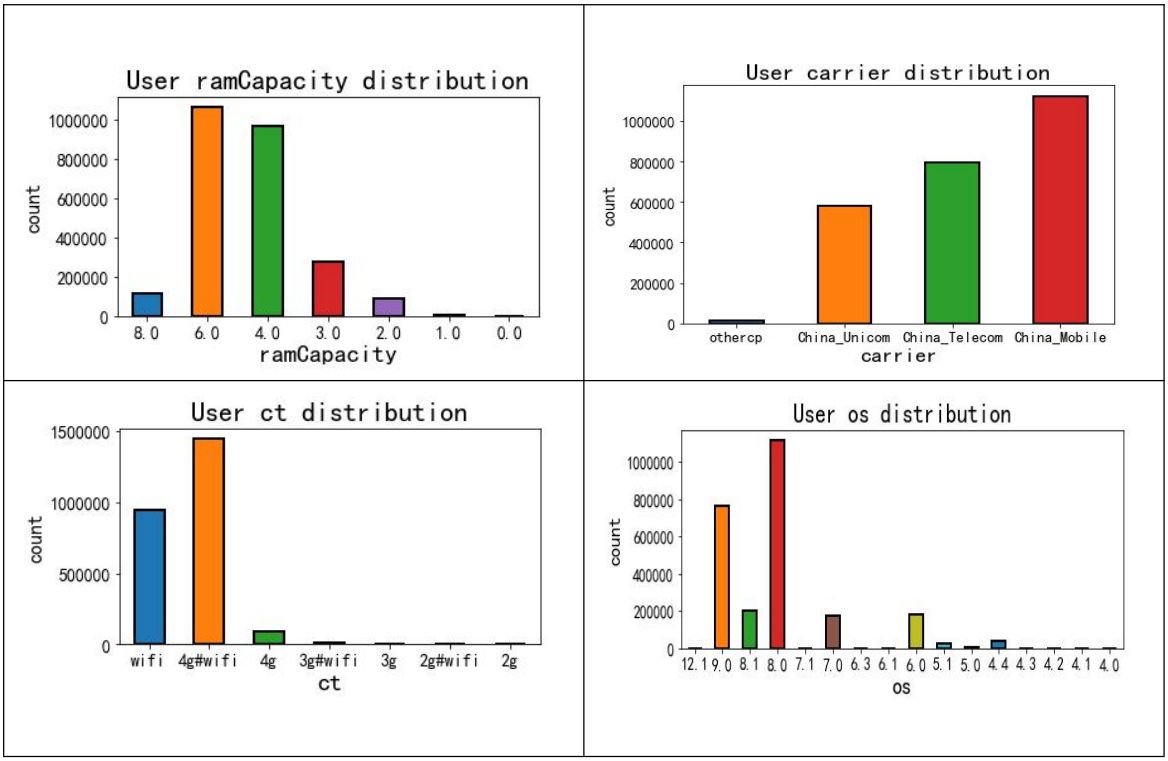


图 5　数据分布单变量统计

- 在其他单变量用户特征中，大部分用户基本信息表中的数据呈离散正态分布情况，主要体现在用户的手机的 RAM 和 ROM 的剩余情况，见图 6。
- 城市、手机颜色、型号主要是离散数据，我们主要进行了随机编码处理。
- 手机字体用户大部分使用的还是默认字体，此处有较大的长尾分布影响情况

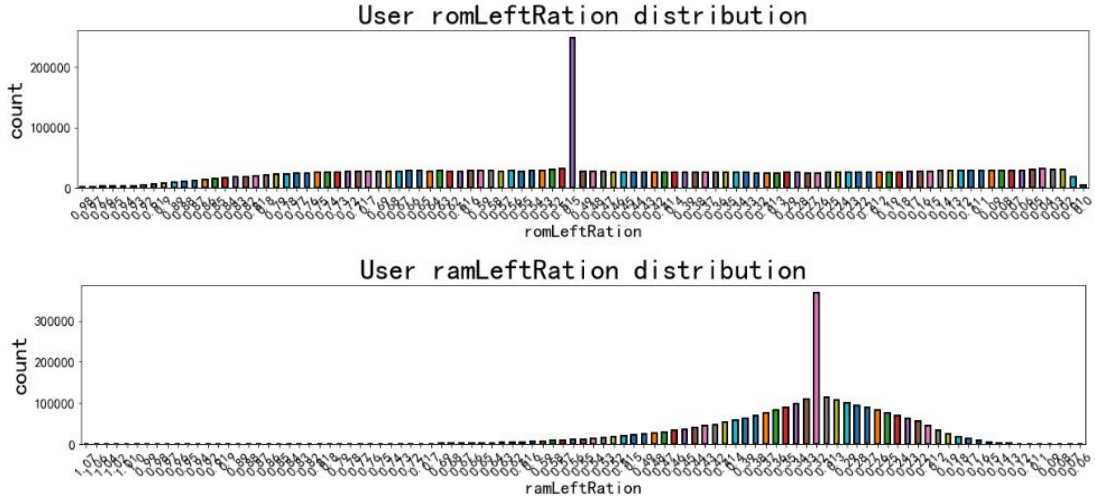


图 6　RAM 和 ROM 的剩余情况

- 用户开机时间：大部分用户在一个月范围内没有一次开机，但是我们也可以通过数据发现有相当部分的用户保持着一定的开关机的周期习惯，其中在 7 次左右是另一个小高峰，除去 0 次开机用户，其他为一个较为标准的正态分布，如图 7。
- 用户行为表中，除了用户开机时间存在比较明显的分布差异，其余的行为基本都呈现比较集中的长尾分布情况，如下图 8，使用次数都集中分布在 0 处，其余选项的样本数量极少。

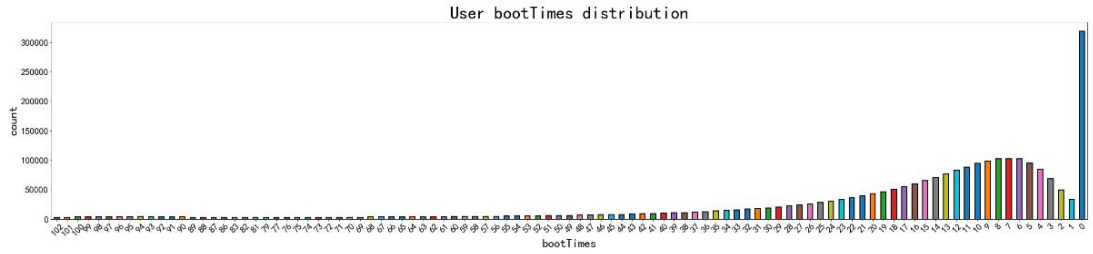


图 7　用户开机次数

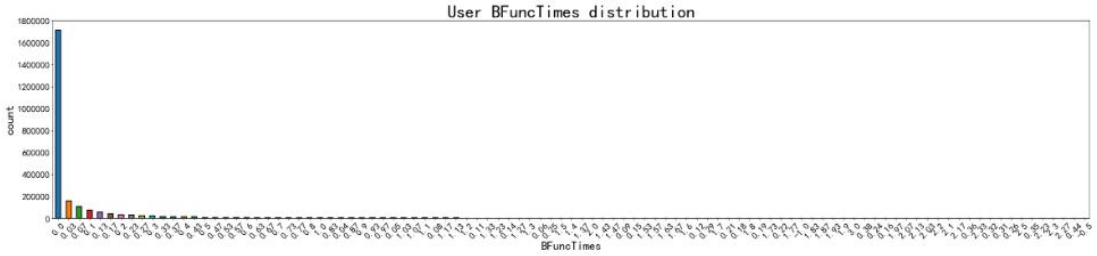


图 8　其余用户行为数据-长尾分布

（4）特征变量&标签关系分析

通过对样本的单变量分析，我们可以了解到数据的样本分布，但是数据样本与年龄标签的之间的联系，需要我们做关联分析才能得到，此处我们利用类似贝叶斯理论中的先验概率相关理论先分析训练集中用户各年龄段的属性以及行为特点。

通过图 2 的标签分布我们可以知道年龄组 3,4,2,5,6,1 人数分别递减，并且分布比例不平衡，故在分析用户行为和属性时应该带着这个比例去观察数据。

- 手机颜色对于用户的选择往往是一个比较大的影响因素，下图 9 列出颜色与训练集年龄组之间的对比分析，可以看到，在不同的年龄组中，不同颜色的分布比例旺旺会存在一些细微差别，**得到不同年龄组选择不同颜色手机的先验概率**。具体我们可以定量分析并定位对应的年龄组与用户群。

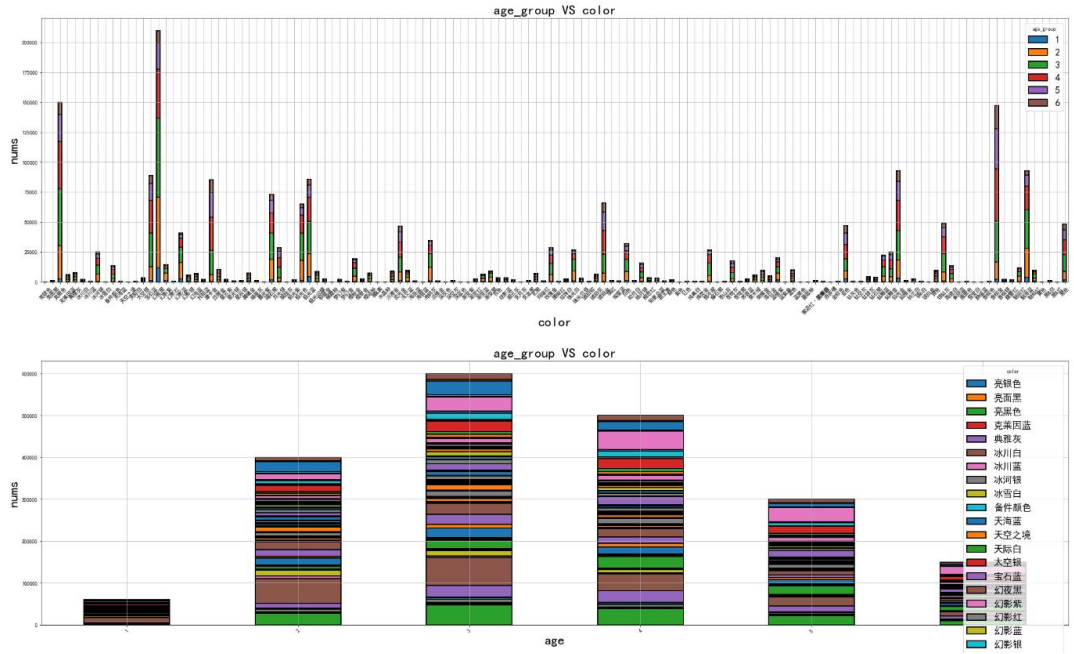


图 9　年龄组与手机颜色的对比

- 除此之外，**手机品牌，字体，系统，RAM，ROM，城市**等信息，我们都可以通过与年龄组的关联，分析出相应年龄组中用户属性和行为的先验概率。另外，以上不同特征之间的**组合**可以唯一定位到一个用户使用的是哪一款**具体配置的手机**，从而精准的将人群进行细分。

3. 特征工程

数据中给出了用户基本信息、行为信息，激活的 app 列表，一个月内 app 的使用情况。以下分别描述团队特征工程的构建方案：

（1）Base_feature 基础特征

- user_basic_info 和 user_behavior_info 构成， prodName, color,City, ct, carrier 分别使用 sklearn.ordinalencoder 做顺序编码
- 类别数量少于 10 个的特征根据算法适用性原则使用独热编码或使用算法模型直接指定类别（如 Lightgbm 和 CatBoost）；
- 手机型号-颜色-字体组合特征（P_C_F）利用先验概率筛选部分类别独热编码
- Ram_Rom_os 组合特征（RRO）利用先验概率筛选部分特色类别独热编码
- PCFRRO 6 个特征组合，利用先验概率筛选部分特色类别独热
- 计算用户 Ram，Rom 剩余量和已经使用的数量 ramLeft,ramUsed,romLeft,romUsed, 并添加相应的排序特征。
- 其他特征根据数据预处理结果直接进行拼接。

（2）App_actived 激活特征

数据使用“#”进行分隔，利用 Python split 方法将其提取，使用 NLP 相关方法处理：

- 使用 sklearn 中的 CountVectorizer 类将用户激活列表数据转为用户安装情况的词频矩阵，【注意】此处转换过程中比较耗费内存，另外词频矩阵中会存在大量的 0 值，建议使用稀疏矩阵形式存储数据，此处我们使用的是 CSR 格式进行存储。
- 使用 sklearn 中的 TfidfVectorizer 类将用户激活列表数据转为用户安装情况的 TF-IDF（词频-逆文本频率）矩阵。
- 使用 gensim 库中的 Word2Vec 训练用户激活列表的词向量表示（此处需要考虑用户安装 APP 的先后顺序，此处赛题没有给出，所以测试效果不佳）
- 按 app 类别排序后，再经过重复采样，组成一个句子，训练 Word2Vec 词向量，目的是体现 app 的类型相似性。
- 关联 app_info 表，获取每个 APP 的类型，统计用户激活的 APP 类型个数
- 统计用户激活 APP 个数
- 用双向 GRU 和 Attention 神经网络训练用户激活列表，输出全连接层之前的隐层特征，即以深度学习的方法，自动提取数据特征。
- 针对每个 app 在训练集中的不同年龄段的安装情况，我们利用先验概率为用户安装的所有 APP 计算出一个 **APP 年龄系数 K**，用于表征该 APP 的对标年龄段，即通过安装比例情况发掘这个 APP 在哪个年龄段更受欢迎，由于训练样本存在不平衡，故不能直接根据安装数量和比例统计，需要做一些特殊处理，添加数据权重的惩罚项。计算方法为：

K = 10*age1_%/0.03+20*'age2_%/0.2+30*age3_%/0.3+40*age4_%/0.25+50*age5_%/0.15 +60*age6_%/0.075

其中，age_n_%代表某个 APP 在 n 年龄段的安装比例，被除数是 n 年龄段在数据集中的样本比例，常系数用以增大不同年龄段区分度，K 为某个的 APP 年龄系数。

（3）App_usage 用户使用特征

- 统计用户使用 APP 在一个月内的使用时长、次数的总量、平均值、方差、最大值、最小值，工作日使用情况、周末使用情况;
- 将用户使用的 app 按照使用时长进行重复采样，使用越久的 app 在列表中的重复次数越多，使 app 列表体现用户对该 app 的使用信息。
- 选取表中出现次数最多的 7000 个 app，形成一个新的 app 列表，并提取该列表的词频特征和 TF-IDF 特征。
- 与激活表类似地，用双向 GRU 和 Attention 神经网络训练用户激活列表，输出全连接层之前的隐层特征。
- 将用户使用的 app 按照使用时长、使用次数和使用日期排序，用 Word2Vec 训练排序后的列表做 Embedding，体现 app 的使用情况相似性。

（4）数据降维

- 将激活表得到的词频矩阵和 TF-IDF 矩阵，用 SVD 方法降维成 300 维。
- 将激活表得到的词频矩阵和 TF-IDF 矩阵，用 LDA（主题模型）方法提取 20 个主题。
- 将使用表得到的词频矩阵和 TF-IDF 矩阵，用 SVD 方法降维成 700 维。

4. 算法建模

（1）数据集的处理

通过线上测试发现，训练集数据与测试集需要预测的数据之间是相同分布的，即训练数据不需要特殊处理，但是在做交叉验证时需要进行相应的分层抽样，保证训练-验证-测试样本的同分布。

（2）算法模型

- **模型的对比**

在比赛前期的整个模型训练过程中，为了尽快验证结果，我们采用 1 折验证集快速验证，发展到比赛中期，我们采用了 5 折交叉验证，均取得了不错的成绩。到了比赛的后期，为了进一步提升模型的准确率，我们采用交叉验证生成 5 折验证集和测试集概率文件用于 Stacking 的第二层的新特征，进行了第二层的训练，进一步提升了线上结果。

在 Stacking 模型选择上我们选取了 LightGBM、XGBoost 和 CatBoost 三种模型分别做相应的测试，并且对三种算法进行不同的性能和使用上的对比：

- 三者都同属树模型，在基础分类器的选择上都可以都参数指定，并且都支持并行训练。
- 在特征处理上，LightGBM 和 CatBoost 可以直接处理类别型特征，而 XGBoost 不支持。
- 在训练时间上，LightGBM 耗时最短，CatBoost 次之，XGBoost 相对训练时间最长。
- 准确率方面，三者在未调参时的预测准确率相差不大。

综合以上，我们主要考虑时间效率为团队成绩提升的首要考虑因素，故在几番测试之后，最终选择神经网络（NN）+LGB 分别做 5 折交叉验证生成 stack 概率特征，使用 LightGBM 做相应的 Stacking。

- **训练与优化**
  - 采用了传统的词袋模型，TfidfVectorizer，Word2Vec 方式提取每个用户的 app 安装情况的词向量，然后采用了 LightGBM 算法对生成的用户安装列表的激活 app 列表向量召开训练和优化。
  - 将激活 app 列表当作一系列的句子，用 NLP 的方法训练模型。对于 RNN 类模型，我们首先需要做词嵌入（Embedding），将句子转换成特征向量表示，经过对比实验，我们选取了 300 维的词向量。
    - RNN 模型一(如图 10)

堆叠（stack）两层双向 GRU，并分别用 Max Pooling、Avg Pooling 和 Attention 方法降维，将三个降维结果合并，最后用 softmax 全连接层进行 6 分类。

- RNN 模型二(如图 11)

双向 GRU 拼接一维卷积形成一个 Block，用两个不同参数的 Block 做拼接，并分别用 Max Pooling、Avg Pooling 和 Attention 方法降维，将三个降维结果合并，最后用 softmax 全连接层进行 6 分类。

- RNN 模型三(如图 12)

使用胶囊网络（CapsNet），出自 Geoffrey Hinton 团队的论文《Dynamic Routing Between Capsules》。输出预测结果，并提取隐层特征作二次使用。

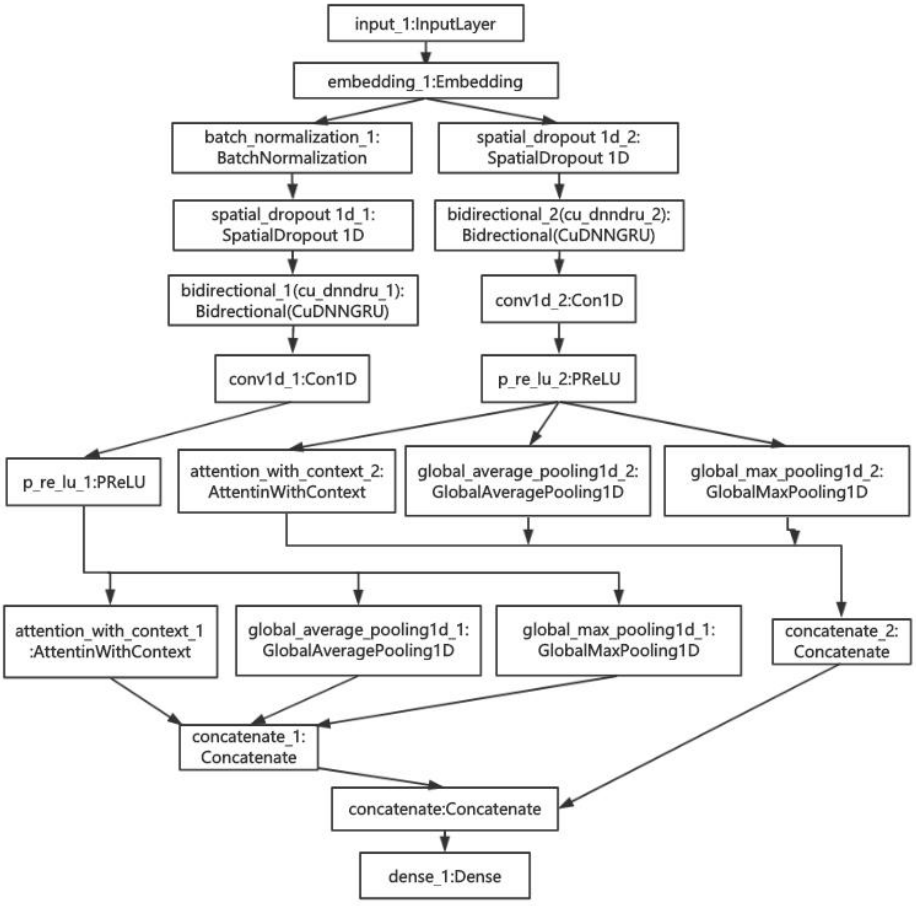


图 10　RNN 模型一示意图

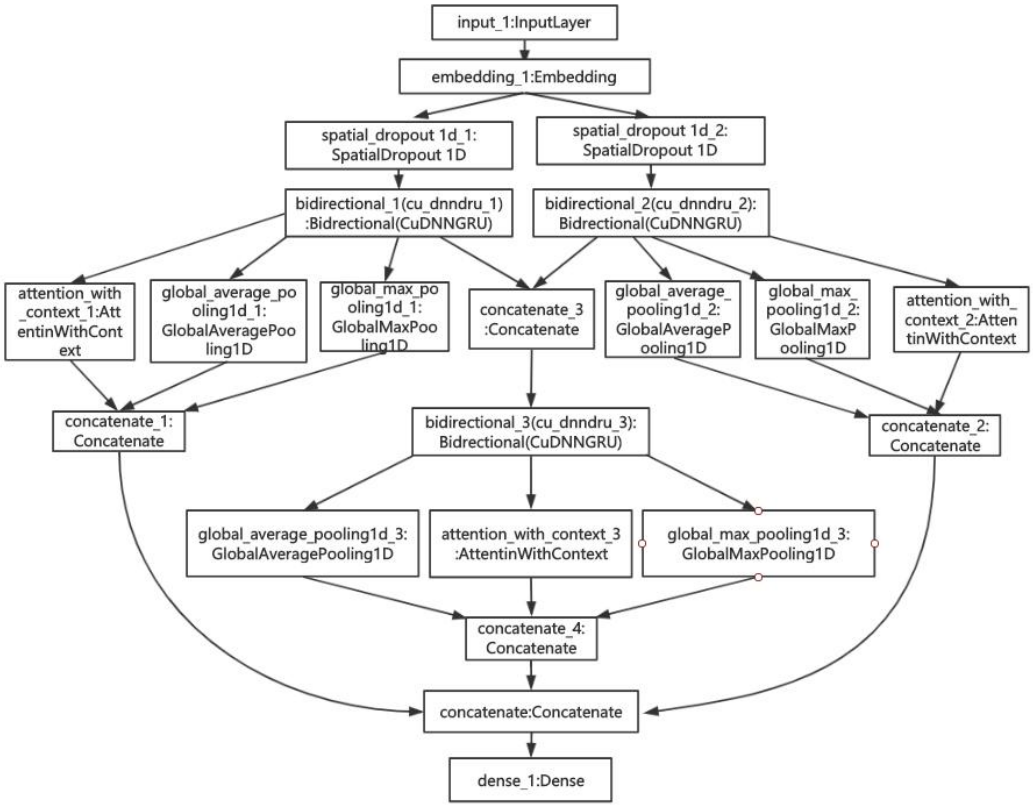


图 11　RNN 模型二示意图

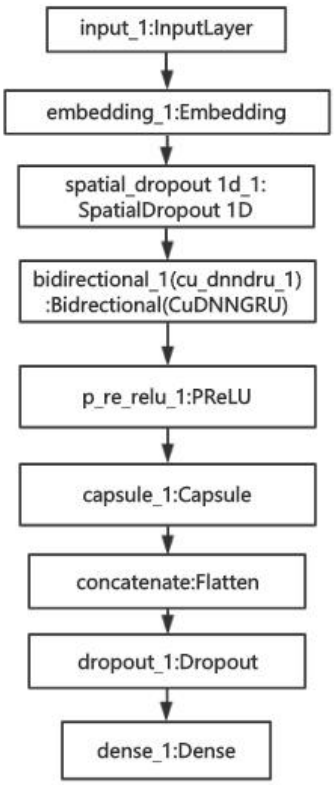


图 12　RNN 模型三示意图

■ 另外，我们还使用了多层感知机（MLP）。MLP 模型设置成三层中间全连接层，每层连接层后都拼接 dropout 层和批标准化层（BatchNormalization），用以防止过拟合。MLP 模型的输入是 app 列表的词频向量或 TF-IDF 向量。MLP 的优势是训练速度快，使用内存或显存空间少，因为不需要对特征向量降维而保留了最多的信息量。

（3）模型融合

- **Stacking**

此处我们将第一层使用 5 个 NN 模型和 4 个 LGB 处理不同特征生成 9 个 stack 概率结果，再使用 LightGBM 做 Stacking 第二层的训练。

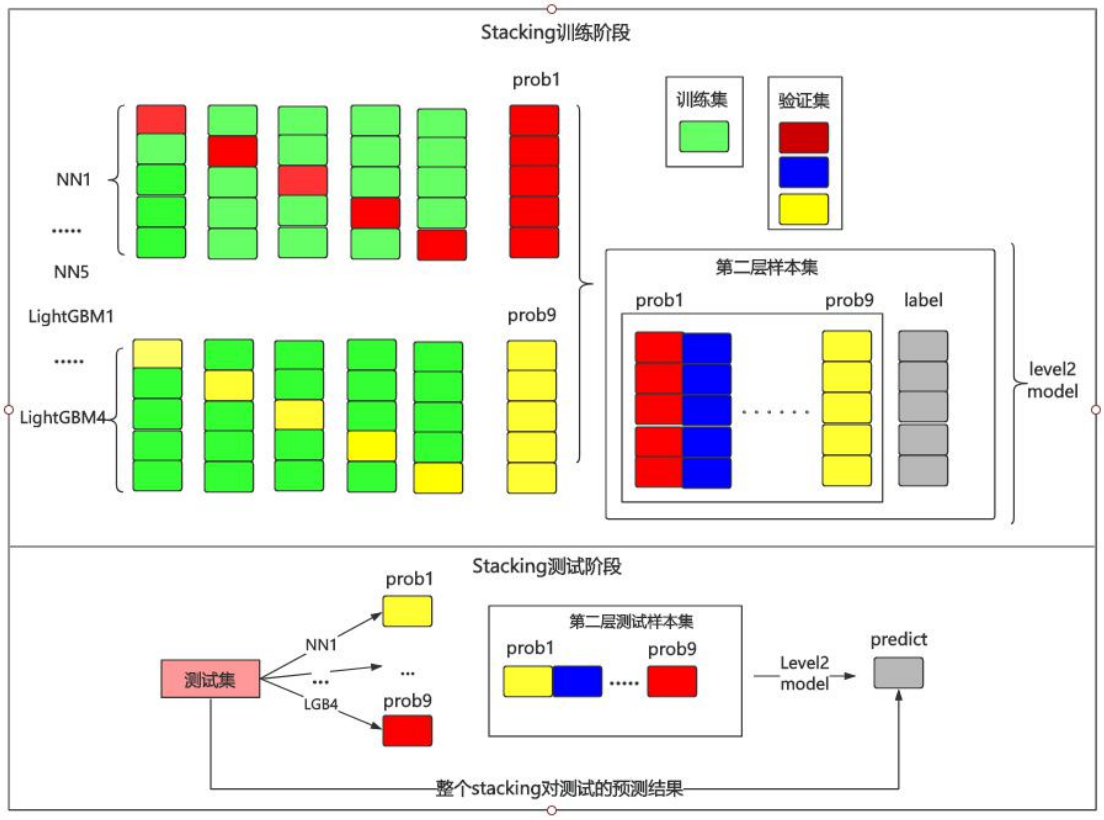


图 13 Stacking 模型融合

- **Bagging**

由 Stacking 生成的概率结果文件作为新的特征进行第二层的训练，此处我们依然使用 LightGBM 作为我们的基础模型，后期通过微调参数训练了两个结果。通过概率结果的 Bagging 线性融合得到最终的提交文件。

5.算法实现说明

（1）描述编译/运行预测代码需要的资源和库以及版本备注；

- 我们的运行环境 Centos7， RAM 256G
- 软件库版本 ：Python3.7, pandas:0.24.2, numpy:1.15.4, matplotlib:3.0.2, sklearn:0.20.3, xgboost:0.90, pickle:4.0， lightgbm:2.2.3,keras:2.2.4,

(2) 实验过程

版本	分数	改进说明
V1	0.42	仅使用两个基础表数据
V2	0.61	+用户激活的 9400 个 APP 的 TF-IDF 特征
V3	0.63	+App_usage 表中挖掘的统计信息，分开预测缺失使用信息的用户，再做两部分样本的拼接
V4	0.64	+RNN 中间层特征
V5	0.6542	+Stacking&Bagging

By: 优生 801

2019 年 7 月 19 日