| | |
|---|---|
| 队伍编号 | dsa2101293 |
| 题号 | （B） |

# 在线教育数据的可视化分析与建模


摘 要

在线教育企业为从众多竞争者中脱颖而出，纷纷致力于拓展互联网获客渠道。然而如何判别高质量的用户和渠道，优化营销成本一直都是各公司的痛点。为此，本小组通过对用户数据开展包括各城市分布情况、登录情况进行多维度可视化分析，发现在线教育 app 的用户量与用户所处城市的经济水平、教育重视程度和人口量相关，本在线教育 app 用户依赖性不足，许多用户仅仅参与了课程，并没有长时间的沉浸学习，表明了线上课程的质量有待提高。然后基于决策树构建用户购买预测模型，所构建模型的查全率为 0.9207，AUC 为 0.935，性能优异，并得到四个与用户是否购买密切相关的变量为领券数量、最后登录距期末天数、关注公众号 1、登录间隔。最后，通过用户流失率分析，发现用户在“开始学习—结课”这一环节的流失率严重；通过相关性分析得到结果，发现 coupon_visit（是否领券访问数）与 coupon（领券数量）具有相关性。令人意外的是，click_buy（购买按钮点击访问数）与 click_notUnlocked（课程未购买弹窗访问数）并不存在相关相关性，一定程度表明未购买弹窗功能在 app 存在的意义并不高。通过对用户的 RFM 分析，将用户分为不同价值的用户，针对不同价值的用户提出不同的营销建议。



关键词：在线教育；可视化；决策树；用户行为分析；RFM 分析

# 目 录

# 1 前言

## 1.1 研究背景

伴随着互联网的迅猛发展,在线教育形式呈现多样化,以其学习成本低等优点被大众接受，因此越来越多创业者投身于在线教育行业蓝海中。在线教育企业为从众多竞争者中脱颖而出，纷纷致力于拓展互联网获客渠道，为公司产品引入新鲜活跃的用户，提高用户购买产品的欲望，提升公司的品牌影响力。然而如何判别高质量的用户和渠道，优化营销成本一直都是各公司的痛点。为此，对用户的行为数据进行分析，判别用户的价值，进而对用户制定专门的营销策略，以实现小成本促销、达到提高用户转化率的目标迫在眉睫。

## 1.2 问题描述

基于以上研究背景，本小组主要完成以下任务：

**任务 1：**获取数据并进行预处理，提高数据质量。

**任务 2：**对用户的各城市分布情况、登录情况进行分析，并分别将结果进行多种形式的可视化展现。

**任务 3：**构建模型判断用户最终是否会下单购买或下单购买的概率，并将模型结果输出为 csv 文件。

**任务 4：**通过用户消费行为价值分析，给企业提出合理的建议。

## 1.3 小组思路

按照任务顺序及内容，本小组确定了总体思路如图 1.1 所示。

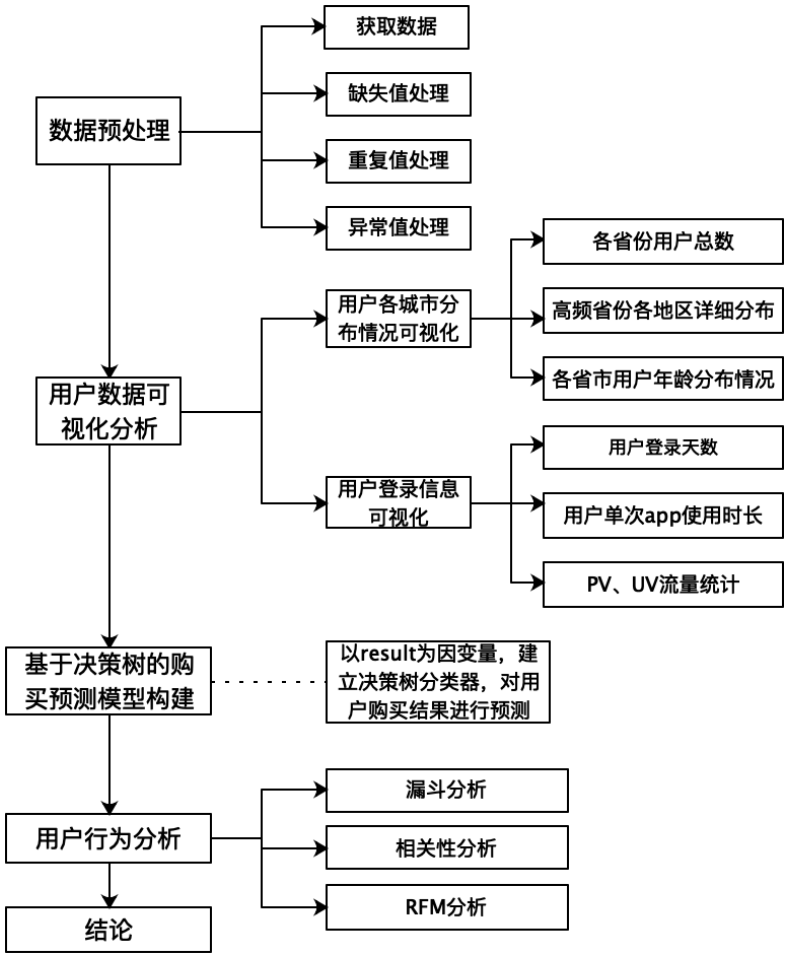


**图 1.1 问题 B 技术路线**

# 2 数据预处理

## 2.1 获取数据

本小组对问题 B 展开分析，首先导入数据进行初步观察。

步骤一：通过 Excel 初识各数据集：user_info.csv、login_day.csv、visit_info.csv 和 result.csv。

步骤二：利用 python 包 pandas 导入各数据集。初步观察发现，user_info 数据集有 135968 个样本，login_day 数据集和 visit_info 数据集各有 135617 个样本，result 数据集有 4639 个样本。

## 2.2 缺失值处理

对各数据集观察后，仅发现 user_info 中 city_num 存在 28209 个缺失值,此处暂不处理，根据后续实际问题选择处理方式。

## 2.3 重复值处理

判断数据集是否存在重复值。发现 4 个数据集的数据均不存在重复值，无须删除。

## 2.4 异常值处理

首先对各数据集进行描述性统计分析，发现数据集 user_info 存在异常，结果如表 2.1 所示。age_month（年龄）的最大值为 24245 个月（约 2020 周岁），这显然不符合常理，而且 age_month 的中位数为 63、下四分位数为 78 与最大值的差距极大。进一步通过对其他字段的分析，本小组判断数据来源于一款主要针对儿童在线教育的产品。因此，本文将 age_month 的数值限制在 144（12 周岁）以内。

**表 2.1 user_info 数据集描述性统计**

| | age_month | platform_num | model_num |
|---|---|---|---|
| count | 135968 | 135968 | 135968 |
| mean | 66.4565 | 10.2504 | 9.41748 |
| std | 209.718 | 1.77566 | 4.44643 |
| min | 0 | 9.2969 | 0 |
| 25% | 51 | 9.2969 | 6.3462 |
| 50% | 63 | 9.2969 | 8.92 |
| 75% | 78 | 9.2969 | 11.4407 |
| max | 24245 | 13.577 | 100 |

另外，通过观察发现，user_info、login_day 以及 visit_info 这三个数据集均存在变量 user_id。然而，user_id 样本量存在差异，考虑到研究的一致性，本文取三个数据集 user_id 的交集，样本数量变为 133935。

我们发现 user_info 数据集中 platform_num（设备）和 model_num（手机型号）为小数，按照我们惯有的认知来说，设备数和收集型号数应该为整数，因此我们将这两个变量的数值进行取整处理。

最后，再对各数据集进行细微考察，在 user_info 数据集的 city_num 变量中，存在异常值 error，我们认为这可能是数据统计不全导致的，因此我们将 error 转化为 NA 值。特别地，在 login_day 数据集中 login_day、login_diff_time、distance_day 存在负值，按照常理来说，登录时间不应存在负数，但我们小组认为-1 代表 0 的含义，因此并未对其进行处理。

# 3 用户数据可视化分析

## 3.1 用户各城市分布情况的可视化

### 3.1.1 各省份用户总数的可视化

用户各城市分布情况，通过数据集 user_info 得到展现，本文需要含有城市信息的用户，剔出了变量 city_num 为 NA 值的样本，最终得到 105769 个样本，同时将这些用户所属地级城市转换为省市，如图 3.1 所示。

统计发现，用户量前三的省市分别为重庆、广东和河南，其中重庆用户量为 12324、广东用户量为 11284 以及河南 8541。本小组认为之所以这三个地区用户量高在于这几个地区对教育的重视程度高，并且人口基数相对较大。

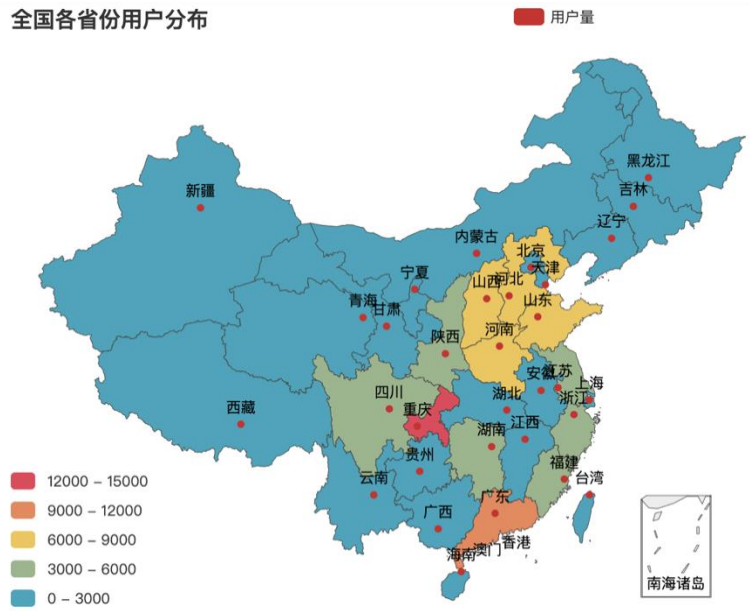


**图 3.1 全国各城市分布情况**

### 3.1.2 高用户省份各地区详细分布情况

本文对用户总量处于前三的省市各地区详细信息进行了统计分析。其中重庆为直辖市无详细区县数据，故本小组最终统计了广东和河南两个地区用户分布情况，如图 3.2、3.3 所示。

统计发现，广东省广州、深圳和东莞地区使用该 app 的用户量最多，本小组认为与这三个地区是广东省经济最发达城市有密切关系。在河南省 m 洛阳、郑州和三门峡使用该 app 用户量最多，三者总量在 60%左右，与广东省类似，河南省这三个地区的经济条件最为突出。

因此本小组得到结论：在线教育 app 的用户集中在经济发达，重视教育和人口基数大的地区。

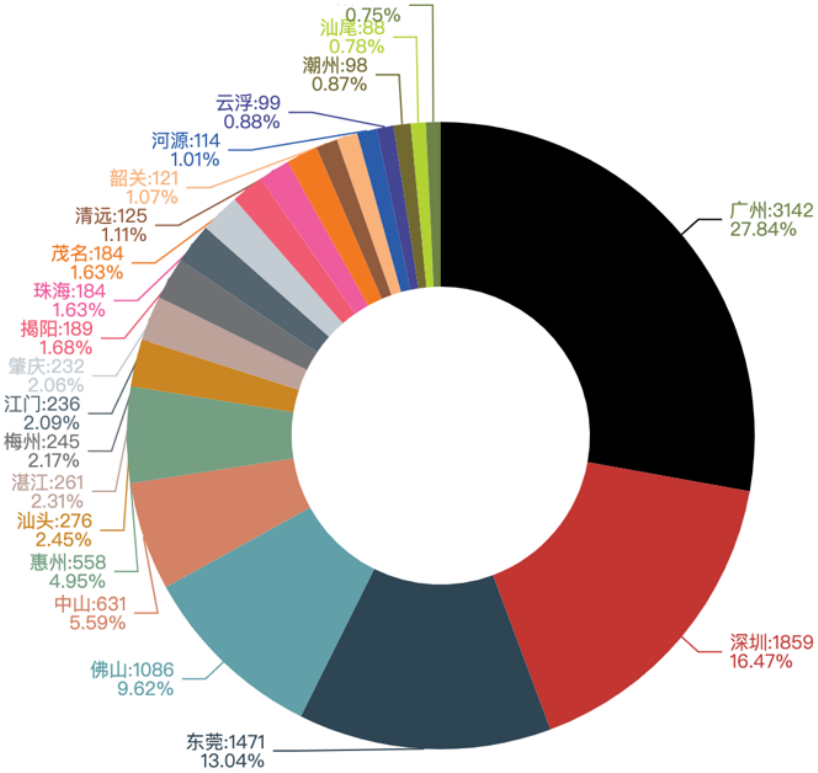


**图 3.2　广东各地区用户量分布**

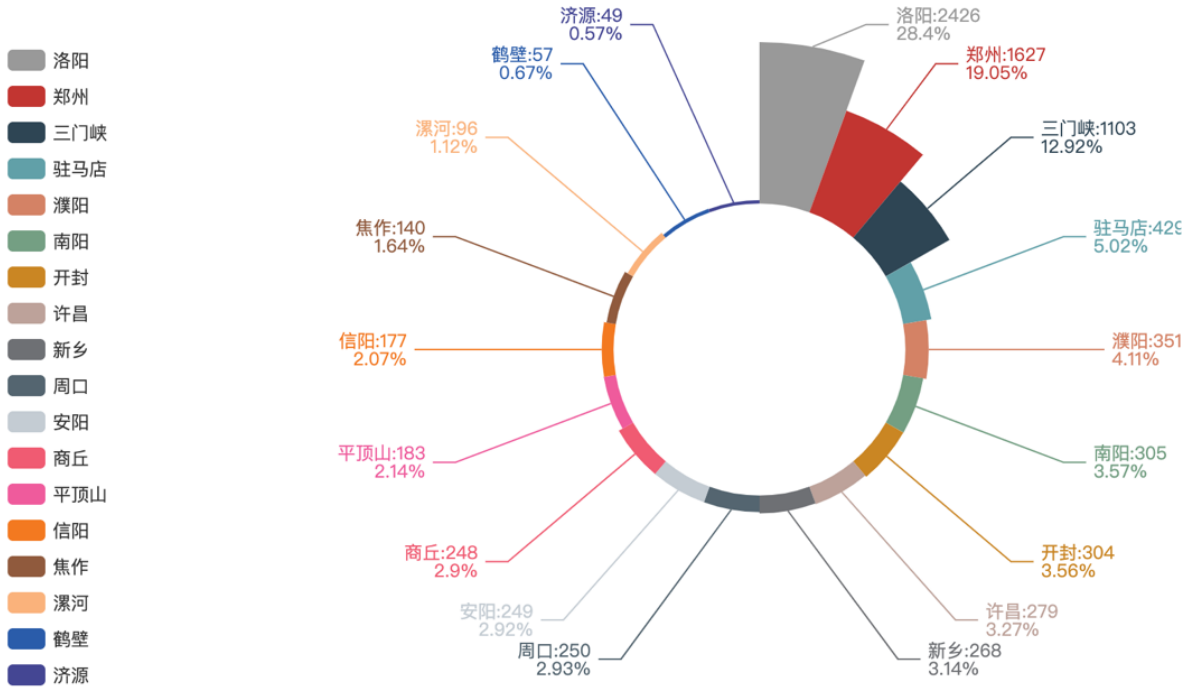


**图 3.3　河南各地区用户量分布**

### 3.1.3　各省市用户年龄分布情况

本文依据 user_info，统计分析了各省市用户平均年龄，如图 3.4 所示。发现各省份用户年龄波动范围不大，平均值在 63.96 月，为 5 岁小朋友。用户年龄与城市用户量并不存在相关关系。

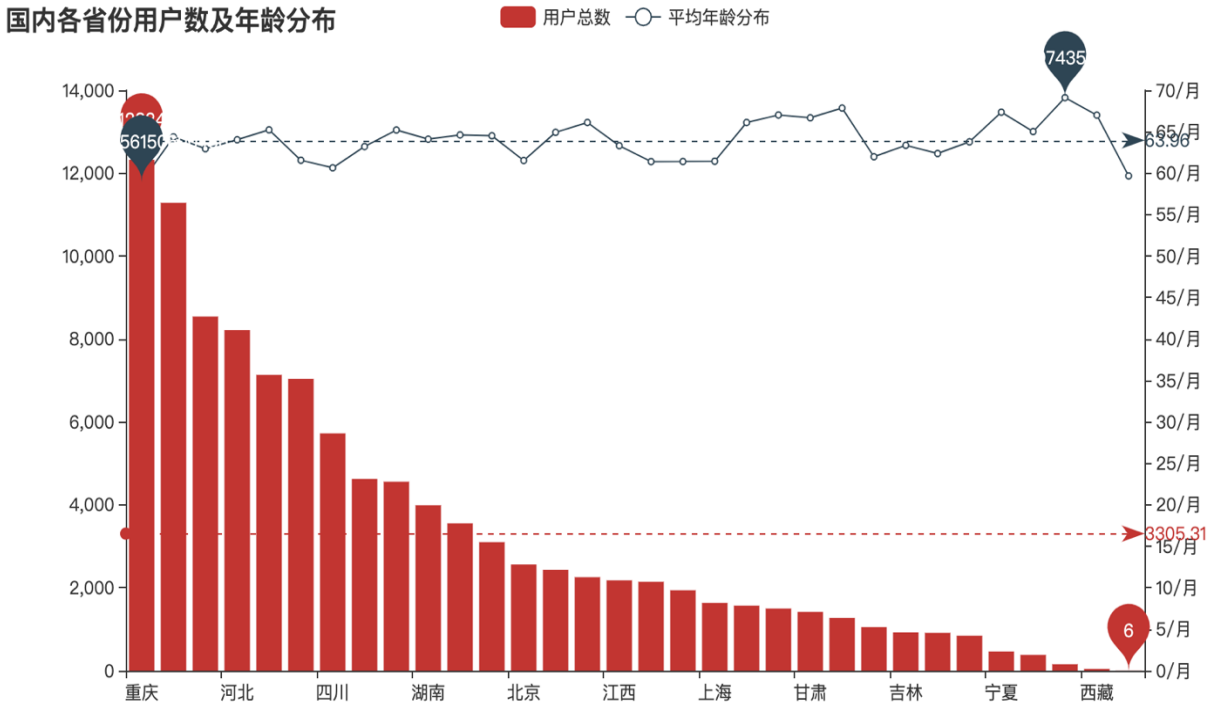


**图 3.4　国内各省份用户数及年龄分布**

## 3.2　用户登录信息可视化

### 3.2.1　用户登录天数可视化

本文对用户登录天数作区间划分为(-6, -3]，(-3, 0]，(0, 3]，(3, 6]，(6, 9]，(9, 20]，(20, 50]，(50, 200],统计如用户登录的天数的玫瑰图 3.5 所示，用户登录天数最多处于(3, 6]中占 45.35%，并且登录天数小于 10 天的用户数达到 95%以上，极少有用户长期使用本 app，这一定程度上产品的用户依赖性不足，进一步可以从产品的使用流程进行多维度分析，寻找原因。

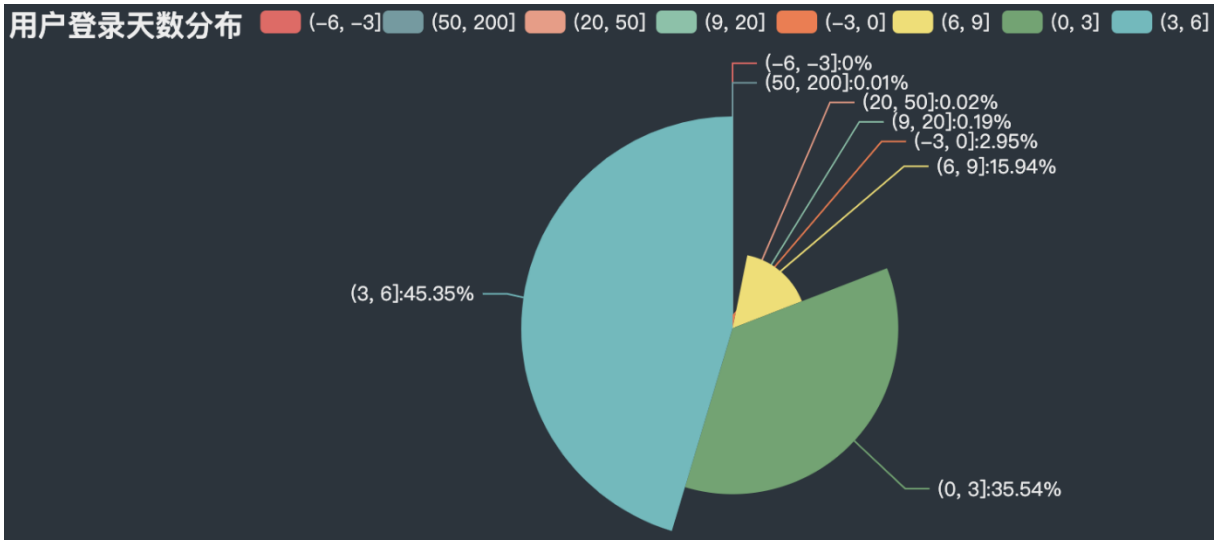


**图 3.5　用户登录天数分布**

### 3.2.2　平均每次登录，用户 app 使用时长可视化

本小组以变量 login_time/login_day 表示平均每次登录用户 app 使用时长，统计如图 3.6 所示。

统计发现用户量随时长增加，先减少后升高再减少。本小组认为 login_time 的单位为分钟，根据单次 app 使用时长统计图可以看出使用时长小于 6 分钟的占总用户数 50%左右，这

表明了许多用户仅仅参与了课程，并没有长时间的沉浸学习，这表明了线上课程的质量有待提高。

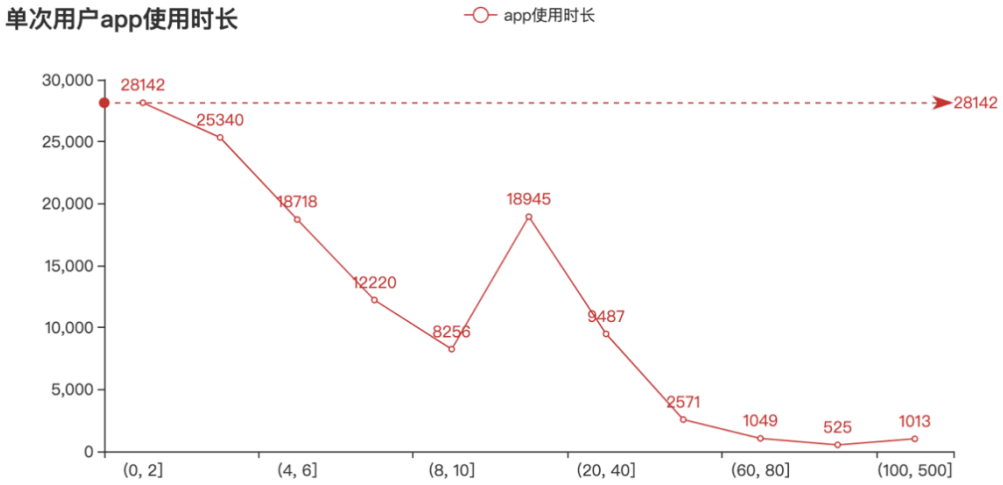


**图 3.6　单次用户 app 使用时长**

### 3.2.3　不同时间 PV、UV 流量统计

PV(Page View)表示页面的浏览量或点击率，其中点击或刷新一次总量增加一次，UV(Unique Visitor)表示独立访客数，1 天内访问某站点的用户数。本小组以变量 first_order_time 为基础统计了 user_id，表示了本 app 的 pv 和 uv，如图 3.7 所示。如图所示，app 的吸引力度有待提高。

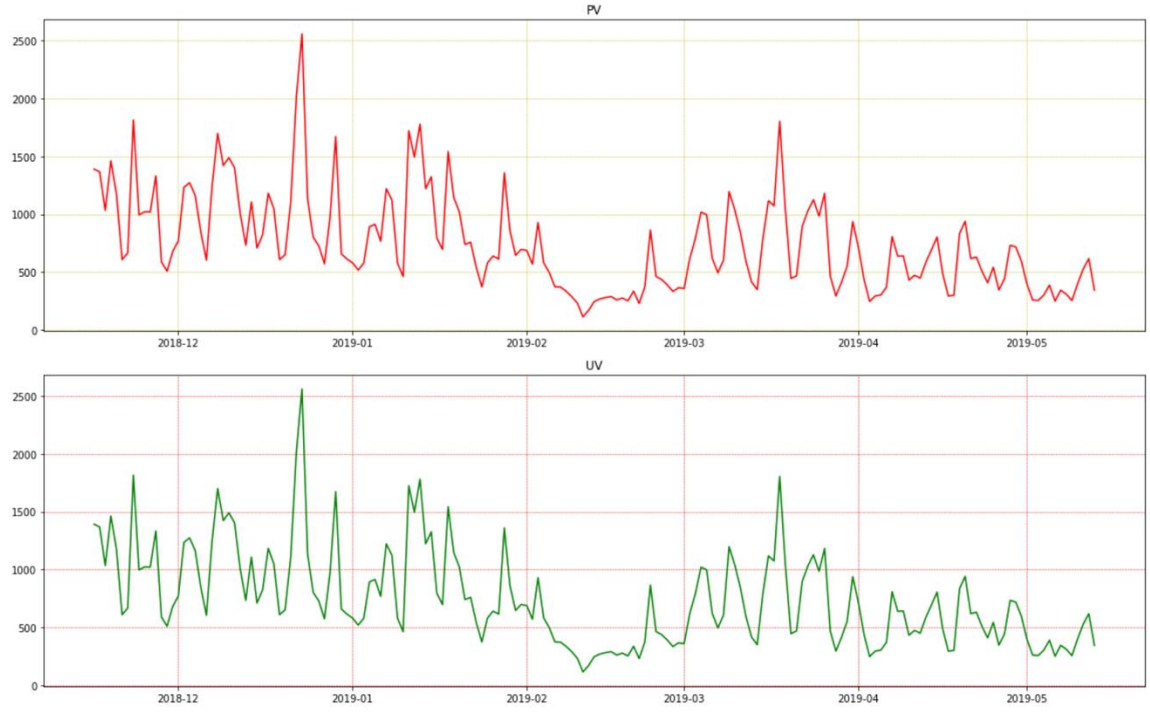


**图 3.7　pv 和 uv**

综上，通过对用户的各城市分布情况、登录情况进行多维度可视化分析，得出以下结论。在线教育 app 的用户量与用户所处城市的经济水平、教育重视程度和人口量相关，另外，本在线教育 app 用户依赖性不足，许多用户仅仅参与了课程，并没有长时间的沉浸学习，表明了线上课程的质量有待提高。

# 4　基于决策树的购买预测模型构建

## 4.1　数据说明

以 result（是否购买：1 代表购买，0 代表不购买）作为因变量，其余变量作为自变量进行建模，其中将体验课下单时间转化为年份，通过模型预测用户是否购买。通过对数据进行预处理，我们得到 133935 条有效样本，但是其中购买的样本数远小于不购买的样本数，故对购买的样本进行随机过采样，最终得到 253935 条有效样本。将处理好的数据随机分成训练集和测试集，其中训练集占总数据的 70%，测试集占总数据的 30%。

## 4.2　决策树分类器

### 4.2.1　建立决策树

首先，使用 R 软件中的 rpart 包，利用决策树对用户是否下单购买进行分类。rpart 函数的选项都用缺省值，其中不纯度度量的方法为 Gini 不纯度。决策树的分类点及分类结果如下图 4.1 所示。

最终，我们得到四个与用户是否购买密切相关的变量，分别是：领券数量、最后登录距期末天数、关注公众号 1、登录间隔。其中领券数量大于等于 1 的用户倾向于下单购买，最后登录距期末天数少的用户倾向于下单购买，关注公众号 1 的用户倾向于下单购买，登录间隔小于 0.82 的用户倾向于下单购买。商家在对用户进行宣传的时候，应该着重对以上类别的用户进行推广，以挖掘潜在客户。

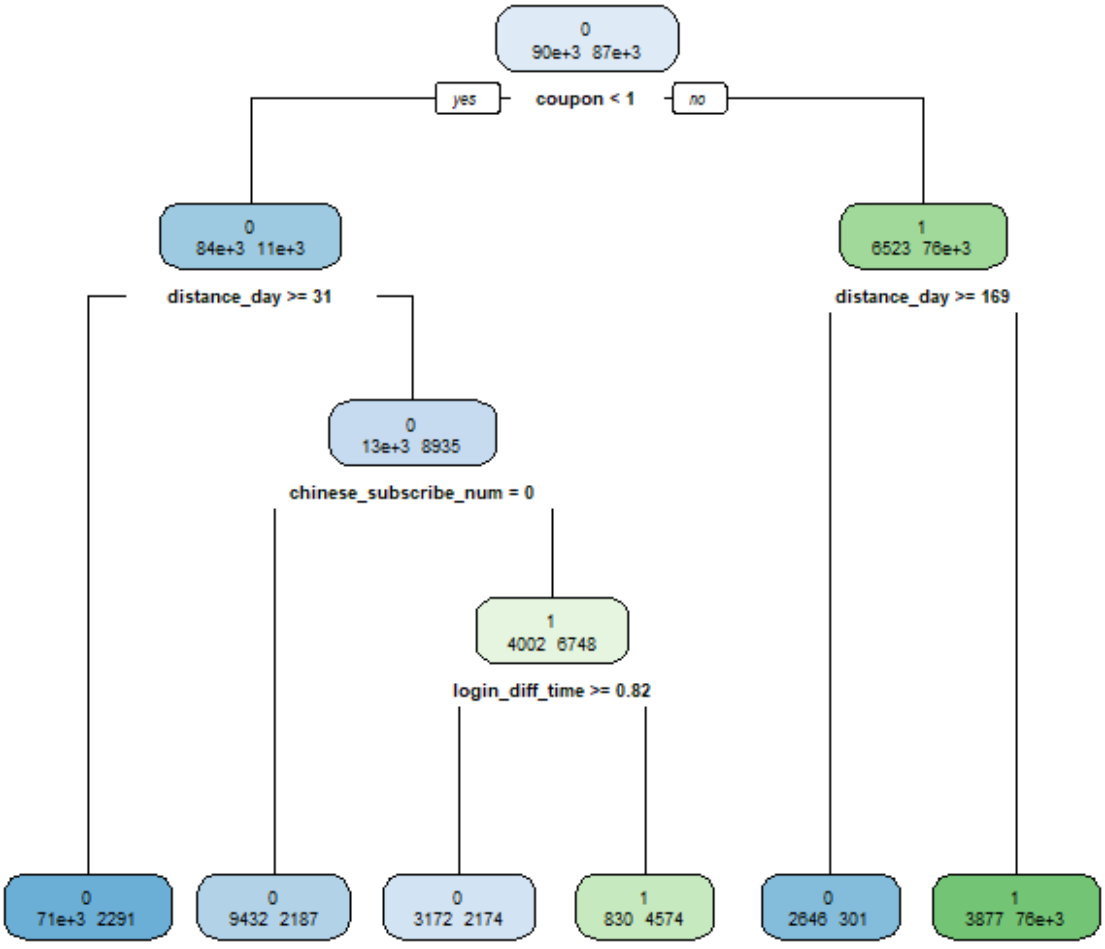


**图 4.1　决策树**

使用单独的决策树做预测，并不是越大越好，因为有可能出现过拟合现象，因此必须限制决策树的规模[1]。程序包 rpart 可以计算一个复杂度参数（cp，默认值为 0.01），用于控制决策树的大小并选择最佳树大小。从表 4.1 中可以看出，第 5 行的 cp（等于 0.01），交叉验证误差较小且相对于其他 cp 值最小，因此，我们得到的决策树不需要进行修剪。

**表 4.1　cp 值及相应的交叉验证误差表**

| | CP | 分割点数 | 交叉验证误差 |
|---|---|---|---|
| **1** | 0.796669 | 0 | 1.00000 |
| **2** | 0.026864 | 1 | 0.20333 |
| **3** | 0.015729 | 2 | 0.17647 |
| **4** | 0.011433 | 4 | 0.14501 |
| **5** | 0.010000 | 5 | 0.13358 |

### 4.2.2　模型评估

对于一个二分类问题，特异性（specificity）代表在所有真是情况为反例的样本中，被预测为反例的比例；查准率（precision）代表在所有预测为正例的样本中，真实情况为正例的比例；查全率（recall）代表在所有真实情况为正例的样本中，被预测为正例的比例；敏感性（sensitivity）=查全率。通过计算公式可以看出，特异性、查准率、查全率越高，模型的预测效果越好。但是，通常情况下，查准率和查全率是一对矛盾的指标，当查准率高时，查全率低；查全率高时，查准率低[2]。在本案例的二分类问题中，下单购买代表正例，未下单购买代表反例。在本案例中，我们应该着重关注查全率，以防止潜在客户流失，因为我们最不希望的是将下单购买的用户预测为不会下单购买的用户。

由表 4.2 中的数据可得，基于训练集的误判率为 0.0655。特异性为 0.948，查全率为 0.9203，查准率为 0.9446。可见，模型在训练集上的预测效果优异。

**表 4.2　基于训练集的混淆矩阵**

| 预测值 / 真实值 | 未下单购买 | 下单购买 |
|---|---|---|
| 未下单购买 | 85757 | 4707 |
| 下单购买 | 6953 | 80338 |

由表 4.3 中的数据可得，基于测试集的误判率为 0.0653。特异性为 0.9463，查全率 0.9227，查准率为 0.9428。可见，模型在测试集上也有优异的预测效果。

**表 4.3　基于测试集的混淆矩阵**

| 预测值 / 真实值 | 未下单购买 | 下单购买 |
|---|---|---|
| 未下单购买 | 36780 | 2088 |
| 下单购买 | 2884 | 34428 |

ROC 曲线是以假正例率（1- specificity）为横坐标，真正例率（sensitivity）为纵坐标，在不同的阈值下对坐标对进行连线，得到的一条曲线，如图 4.2 所示。ROC 曲线上一个好的分类器应该紧贴左上角，曲线上离左上角最近的点（0.927，0.943）对应的临界点（1.5）即为使得该分类器最好的临界点。AUC 是 ROC 曲线下方的面积，代表根据当前的分类器输出值进行排序，正例排在反例前面的概率。显然，AUC 的值越大越好。在本案例中，AUC=0.935，代表模型预测效果优异。

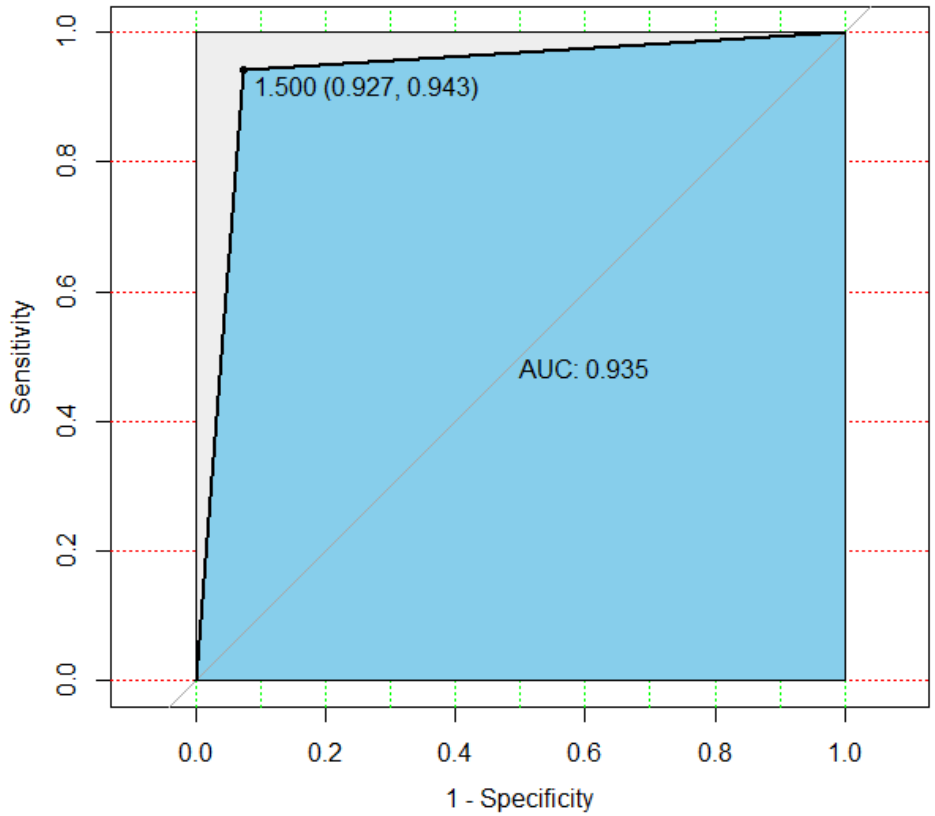


**图 4.2　ROC 曲线**

### 4.2.3 结果预测

利用我们经过过采样得到样本训练的分类器对原有的 133935 条有效样本进行预测，预测结果见（predict_result_output.csv），其误判率为 0.0535，混淆矩阵如表 4.4 所示。在对用户是否下单进行预测时，我们最不希望的是将下单购买的用户预测为不会下单购买的用户。而本分类器进行对原有的 133935 条有效样本预测时，查全率为 0.9207，此分类器可以有效的识别潜在客户。

**表 4.4　基于原始样本的混淆矩阵**

| 预测值 / 真实值 | 未下单购买 | 下单购买 |
|---|---|---|
| 未下单购买 | 122537 | 6795 |
| 下单购买 | 365 | 4238 |

# 5　用户行为分析

## 5.1　用户流失率分析—漏斗分析模型

### 5.1.1　漏斗分析方法简介

从业务流程起点开始到最后目标完成的每个环节都会有用户流失，漏斗分析方法就是用来衡量业务流程每一步的转化效率的分析方法[3]。它有两个指标，分别是环节转化率和总体转化率。环节转化率=本环节用户数/上一环节用户数，是为了衡量相邻业务环节的转化情况。总体转化率=某环节用户数/第 1 环节用户数，是为了衡量从第 1 环节到该环节为止总体的转化情况。

### 5.1.2　漏斗分析方法的作用

漏斗分析的作用是“定位问题节点”，即找到出问题的业务环节在哪。经过各个业务环节转化下来的用户，会产生更大的价值。因为这部分用户更加忠诚，更认可业务的流程。随着转化用户的不断增加，留存用户的规模也在不断增加，产品的盈利规模也会随之增加。

### 5.1.3　漏斗分析模型

针对这个项目，将业务流程分为四个部分：用户注册 app、用户开课加入课程、用户学习课程、完成课程。在计算完成课程数时，我们通过画完成课程数-用户人数的折线图，发现当 finish_num=5 是一个转折点，因此我们将 finish_num>=5 作为判断用户结课的标志。

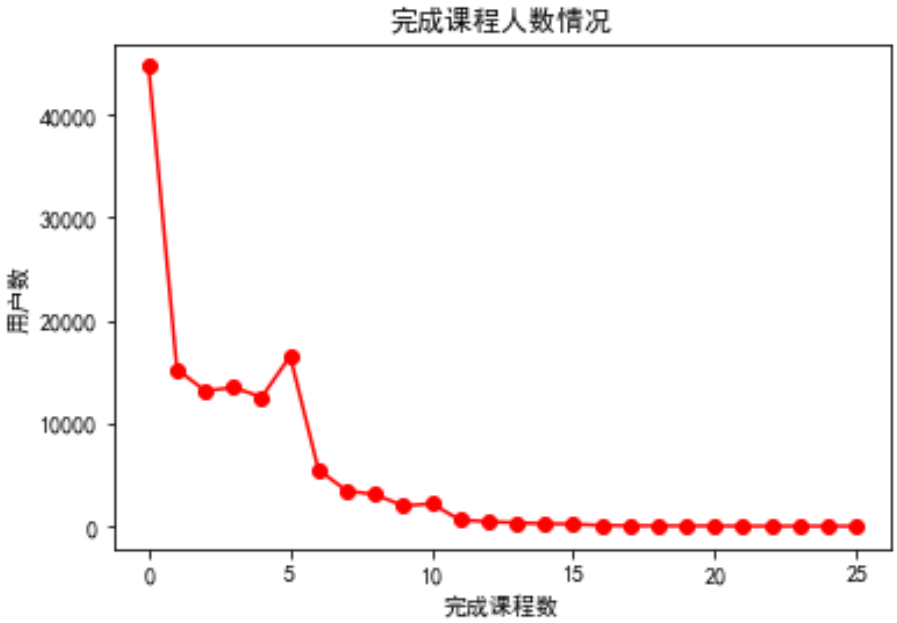


**图 5.1　用户完成课程人数**

计算总体转化率和各环节转化率，结果如表 5.1 所示。分别将总体转化率和环节转化率用漏斗图展示出来，如图 5.2 和 5.3 所示。

**表 5.1　总体转化率和各环节转化率**

| 环节 | 人数 | 总体转化率(%) | 环节转化率(%) |
|---|---|---|---|
| 注册 | 133935 | 100 | 100 |
| 加入课程 | 124367 | 92.86 | 92.86 |
| 开始学习 | 107406 | 80.19 | 86.36 |
| 学完 5 节课程 | 34726 | 25.93 | 32.33 |

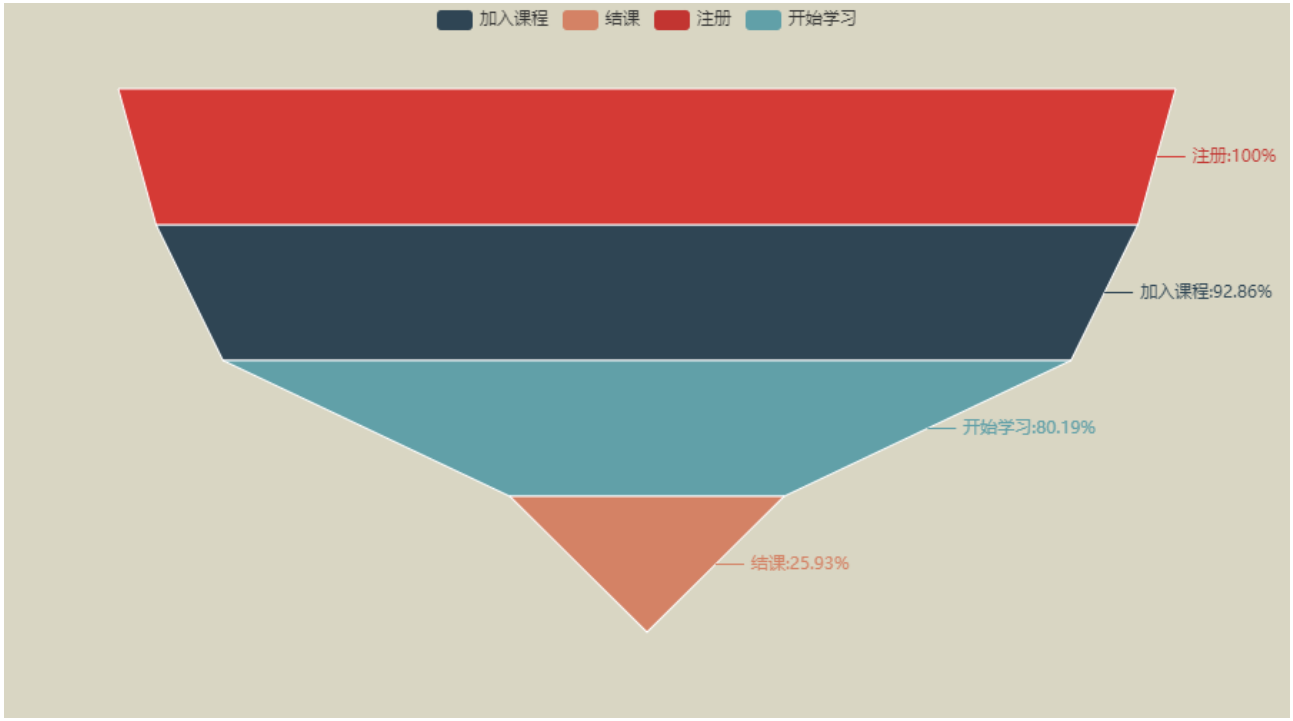


**图 5.2　总体转化率漏斗图**

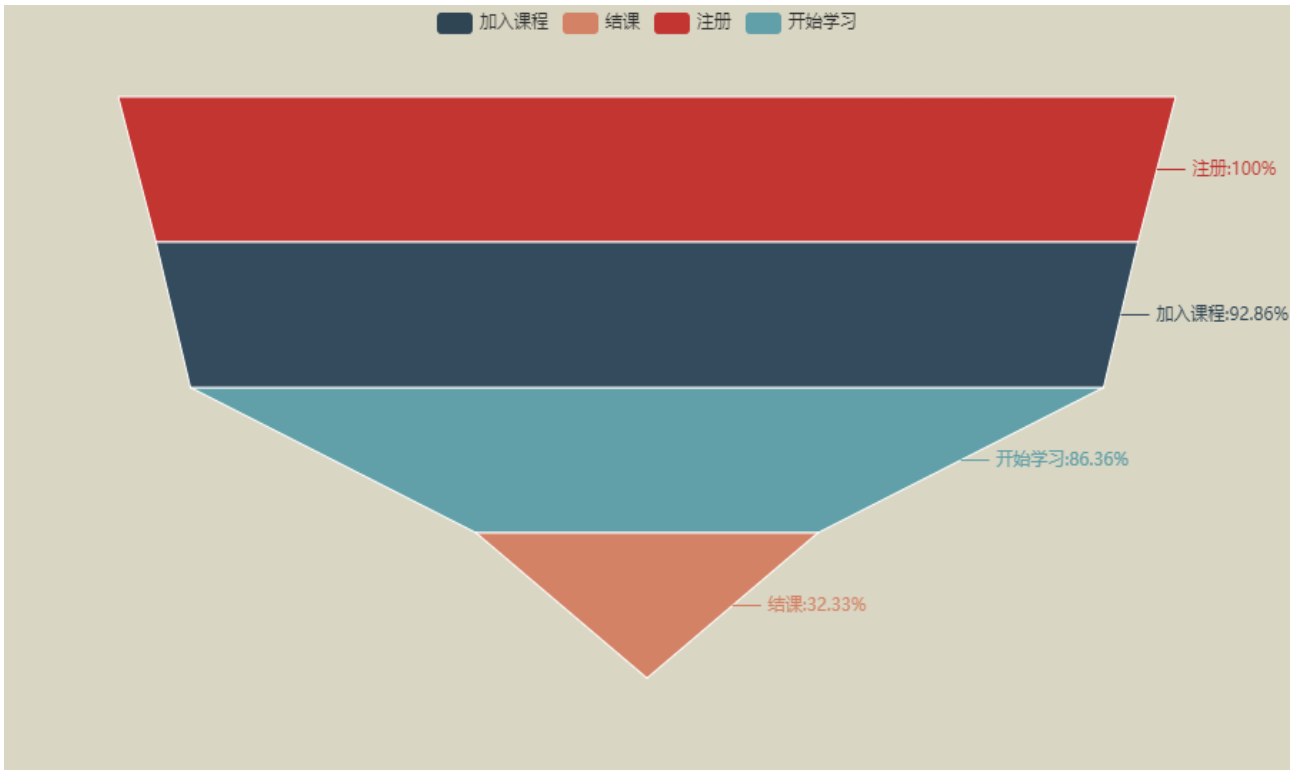


**图 5.3　环节转化率漏斗图**

根据上述的漏斗图，我们发现环节转化率最低的为“开始学习—结课”（转化率为 32.33%），这意味着很多用户开始学习了课程，但并没有长期坚持学习。我们通过图 5.1 的折线图可以发现，在学完 5 节课之后，继续学习的人数急剧下降。因此，企业应该采取措施，着重提高“开始学习—结课”这一环节的转化率。比如，采取积分兑换、打卡式学习等用户激励体系，激励用户完成课程学习，做好用户运营[4]。

## 5.2 用户购买行为与领券行为的相关性分析

用户购买行为与领券行为是用户消费价值行为分析的一个重要方面，本小组对用户购买行为与领券行为进行了相关性分析。

新建 consumer_behavior 数据集包含变量 user_id、first_order_price、coupon、coupon_visit、click_buy、click_notUnlocked 应用于相关性分析。对数据集 consumer_behavior 数据进行标准化处理。

数据标准化方法：

数据归一化通过将变量转化为[0，1]间的数以剔除单位及量纲对评价过程的影响。设：$x_{ij}$ 表示第 i 个 app 用户第 j 个变量归一化值；$p_{ij}$表示第 i 个 app 用户第 j 个变量的数据；n 表示样本总数，公式如式 1 所示[5]。

$$x_{ij} = \frac{p_{ij} - \min\limits_{1\leqslant i\leqslant n}(p_{ij})}{\max\limits_{1\leqslant i\leqslant n}(p_{ij}) - \min\limits_{1\leqslant i\leqslant n}(p_{ij})} \quad (1)$$

相关性分析方法：

计算变量间的相关系数。设：$r_{qj}$－第 q 和第 j 个变量的相关系数；$x_{iq}$－第 i 个用户第 q 个指标值；$\bar{x}_q$－第 q 个变量平均值；$x_{ij}$－第 i 个用户第 j 个指标值；$\bar{x}_j$－第 j 个指标平均值。则$r_{qj}$为[6]：

$$r_{qj} = \frac{\sum_{i=1}^{n}(x_{iq}-\bar{x}_q)(x_{ij}-\bar{x}_j)}{\sqrt{\sum_{i=1}^{n}(x_{iq}-\bar{x}_q)^2 \sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}} \quad (2)$$

带入数据得到表 5.2：

**表 5.2 consumer_behavior 相关性分析结果**

| | user_id | coupon_visit | click_buy | click_notunlocked | first_order_price | coupon |
|---|---|---|---|---|---|---|
| user_id | 1 | -0.0661 | -0.00744 | -0.02807 | -0.01223 | -0.06869 |
| coupon_visit | -0.0661 | 1 | -0.00052 | 0.102558 | 0.035616 | 0.515299 |
| click_buy | -0.00744 | -0.00052 | 1 | 0.003178 | -0.00051 | 0.009889 |
| click_notunlocked | -0.02807 | 0.102558 | 0.003178 | 1 | 0.007767 | 0.056421 |
| first_order_price | -0.01223 | 0.035616 | -0.00051 | 0.007767 | 1 | 0.007185 |
| coupon | -0.06869 | **0.515299** | 0.009889 | 0.056421 | 0.007185 | 1 |

根据得到的相关性结果，发现 coupon_visit（是否领券访问数）与 coupon（领券数量）相关系数为 0.51，具有相关性。令人意外的是，click_buy（购买按钮点击访问数）与 click_notUnlocked（课程未购买弹窗访问数）并不存在相关相关性，一定程度表明未购买弹窗功能在 app 存在的意义并不高。

## 5.3 用户价值分析—RFM 模型

### 5.3.1 RFM 方法简介

RMF 是 3 个指标的缩写：最近一次消费时间间隔（Recency）、消费频率（Frequency）、消费金额（Monetary），通过这 3 个指标对用户进行分类的方法称为 RFM 分析方法。把这 3 个指标按照价值从低到高排序，就可以得到用户分类的规则，如表 5.3 所示。我们可以对不同价值的用户类型使用不同的运营决策，把公司有限的资源最大化。

表 5.3　用户分类规则

| 用户分类 | 最近一次消费时间间隔（R） | 消费频率（F） | 消费金额（M） |
|---|---|---|---|
| 1. 重要价值用户 | 高 | 高 | 高 |
| 2. 重要发展用户 | 高 | 低 | 高 |
| 3. 重要保持用户 | 低 | 高 | 高 |
| 4. 重要挽留用户 | 低 | 低 | 高 |
| 5. 一般价值用户 | 高 | 高 | 低 |
| 6. 一般发展用户 | 高 | 低 | 低 |
| 7. 一般保持用户 | 低 | 高 | 低 |
| 8. 一般挽留用户 | 低 | 低 | 低 |

### 5.3.2　RFM 分析

首先，我们将用户体验课下单时间的分布图画出来，如图 5.4 所示。通过折线图我们发现，在 2018 年 12 月—2019 年 1 月这个时间段下单的人数较多，因此我们我们以 2018 年 12 月 31 日为时间节点，分析在 2018 年 12 月下单的用户行为。

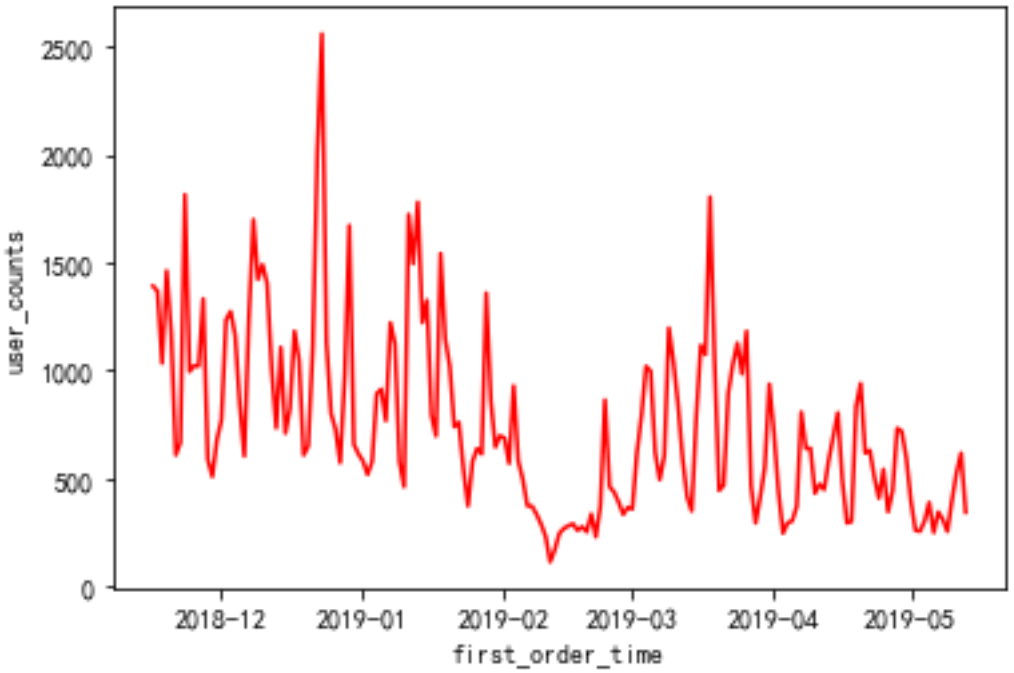


图 5.4　用户下单时间分布图

我们发现原始数据集 user_info 中的每个用户都是唯一的，也就是说每个用户只下单了一次。也就是说，在 RFM 中 F 的值全部为 1，这对我们的分析没有任何价值，因此，在本项目中我们不考虑 F 值。我们先计算 R 和 M 值，然后对不同的 R、M 值进行打分，我们认为#打分标准为：R<3 记 5 分，R 为 3～5 记 4 分，R 为 5～10 记 3 分，R 为 10～20 记 2 分，R>20 记 1 分； M>120 记 5 分，M 为 90～120 记 4 分，M 为 50～90 记 3 分，M 为 10～50 记 2 分，M 为 10 以内记 1 分。根据打分规则，我们将 R、M 值进行分类。其次，分别计算 R、M 值的平均值，若 R、M 值高于各自的平均值，则将“R 值高低”、“M 值高低”各自记为“高”；反之，则将“R 值高低”、“M 值高低”各自记为“低”。最后根据用户分类规则将用户分为 4 类，计算每一类用户的总人数。结果如表 5.4 所示。

通过表 5.4 的结果发现，我们研究的用户几乎全部都是一般用户，仅仅只是 16 个人被划分为重要用户，这个结果令人震惊。因此，企业应该采取措施提高重要用户的占比情况，比

如将 App 的界面变得更加美观以吸引更多的客户，以及企业应该注重线上教学的质量来保留重要价值用户等等。具体的，针对重要发展用户，企业应该开发更多高价值的课程，提高消费频次；针对重要挽留用户，企业应该重点联系，开发用户感兴趣的课程并进行多次推荐，提高用户的留存率；针对一般发展用户，发送多种定向优惠券，增加购买频次和总消费金额；针对一般挽留用户（在本项目中占比最高），企业应该发掘用户兴趣，使用消息推送等低成本手段触达和挽留[3]。

**表 5.4　用户最终分类**

| R 值高低 | M 值高低 | 用户分类 | 总人数 |
|---|---|---|---|
| 高 | 高 | 重要发展用户 | 4 |
| 低 | 高 | 重要挽留用户 | 12 |
| 低 | 高 | 一般发展用户 | 11719 |
| 低 | 低 | 一般挽留用户 | 22106 |

# 6 结论

1．通过对用户的各城市分布情况、登录情况进行多维度可视化分析，得出以下结论。在线教育 app 的用户量与用户所处城市的经济水平、教育重视程度和人口量相关，另外，本在线教育 app 用户依赖性不足，许多用户仅仅参与了课程，并没有长时间的沉浸学习，表明了线上课程的质量有待提高。

2．以 result（是否购买：1 代表购买，0 代表不购买）作为因变量，其余变量作为自变量进行建模，预测用户是否会下单购买。模型的查全率为 0.9207，AUC 为 0.935，表明模型预测性能优异。我们得到四个与用户是否购买密切相关的变量，分别是：领券数量、最后登录距期末天数、关注公众号 1、登录间隔。其中领券数量大于等于 1 的用户倾向于下单购买，最后登录距期末天数少的用户倾向于下单购买，关注公众号 1 的用户倾向于下单购买，登录间隔小于 0.82 的用户倾向于下单购买。商家在对用户进行宣传的时候，应着重对以上类别的用户进行推广，以挖掘潜在客户。

3．通过对用户流失率的分析，我们发现在“开始学习—结课”这一环节用户的转化率最低，因此企业可以采取用户激励体系来激励用户完成学习，比如积分兑换、课程打卡等方式都是不错的方法。通过相关性分析得到结果，发现 coupon_visit（是否领券访问数）与 coupon（领券数量）相关系数为 0.51，具有相关性。令人意外的是，click_buy（购买按钮点击访问数）与 click_notUnlocked（课程未购买弹窗访问数）并不存在相关相关性，一定程度表明未购买弹窗功能在 app 存在的意义并不高。通过对用户价值的 RFM 分析，我们发现在本项目中，重要用户所占的比重极少，因此企业应该着重对重要发展用户和重要挽留用户采取相应的措施。比如，针对重要发展用户，企业应该开发更多高价值的课程，提高消费频次；针对重要挽留用户，企业应该重点联系，开发用户感兴趣的课程并进行多次推荐，提高用户的留存率。

# 参考文献

[1] 吴喜之. 数据科学导论：R 与 python 实现[M]. 高等教育出版社，2019:163.
[2] 周志华. 机器学习[M]. 清华大学出版社，2016:30.
[3] 猴子•数据分析学院. 数据分析思维：分析方法和业务知识[M]. 清华大学出版社，2020.
[4] 在线教育 APP 运营:如何激励用户完成课程学习[J]，信息与电脑，2017(19):4-5.
[5] 周颖,沈隆. 基于 Brown-Mood 中位数检验的小企业债信评级体系[J]. 系统管理学报，2020，29(06)：22－34.
[6] 于善丽，迟国泰，姜欣. 基于指标体系违约鉴别能力最大的小企业债信评级体系及实证[J]. 中国管理科学，2020，28(06)：38－50.

# 附录

**#任务 1 ：获取数据并进行预处理，提高数据质量**

```
#加载程序包
import pandas as pd
import numpy as np
from pyecharts.charts import *
from pyecharts import options as opts
import matplotlib as    mpl
from matplotlib import pyplot as plt
import seaborn as sns
import openpyxl
#导入数据
user_info=pd.read_csv('user_info.csv')
login_day=pd.read_csv('login_day.csv')
visit_info=pd.read_csv('visit_info.csv')
result=pd.read_csv('result.csv')
print(user_info.info())
print(login_day.info())
print(visit_info.info())
print(result.info())
# 判断是否存在缺失值
print(user_info.isnull().sum())
print(login_day.isnull().sum())
print(visit_info.isnull().sum())
print(result.isnull().sum())
# 判断是否存在重复值，若存在剔除重复值
print(user_info.duplicated().sum())
print(login_day.duplicated().sum())
print(visit_info.duplicated().sum())
print(result.duplicated().sum())
print(user_info.describe())
print(login_day.describe())
print(visit_info.describe())
print(result.describe())
user_info = user_info[user_info['age_month'] <= 144]     # user 年龄限制在 12 周岁内
user_info = user_info[user_info['user_id'].isin(login_day['user_id'])]   # 统一 user_id
visit_info = visit_info[visit_info['user_id'].isin(user_info['user_id'])]   # 统一 user_id
login_day = login_day[login_day['user_id'].isin(visit_info['user_id'])]   # 统一 user_id
user_info['platform_num'] = user_info['platform_num'].apply(lambda x:round(x))
user_info['model_num'] = user_info['model_num'].apply(lambda x:round(x))
user_info['city_num'][user_info['city_num']=='error'] = np.nan
```

**#任务 2 ：对用户的各城市分布情况、登录情况进行分析，并分别将结果进行多种形式的可视化展现**

```
user_info1 = user_info.dropna(subset=['city_num'])
user_info1 = user_info1.reset_index(drop=True)
```

```
print(user_info1)
# 将城市转换为省份,并保存至 province.csv
area_data = {
        '北京':['北京','朝阳区', '海淀区', '通州区', '房山区', '丰台区', '昌平区', '大兴区', '顺义区', '西城区', '延庆县', '石景山区', '宣武区', '怀柔区', '崇文区', '密云县',
                '东城区', '门头沟区', '平谷区'],
        '广东':['广东', '东莞', '广州', '中山', '深圳', '惠州', '江门', '珠海', '汕头', '佛山', '湛江', '河源', '肇庆','潮州', '清远', '韶关', '揭阳', '阳江', '云浮', '茂名', '梅州', '汕尾'],
        '山东':['山东', '济南', '青岛', '临沂', '济宁', '菏泽', '烟台','泰安', '淄博', '潍坊', '日照', '威海', '滨州', '东营', '聊城', '德州', '莱芜', '枣庄'],
        '江苏':['江苏', '苏州', '徐州', '盐城', '无锡','南京', '南通', '连云港', '常州', '扬州', '镇江', '淮安', '泰州', '宿迁'],
        '河南':['河南', '郑州', '南阳', '新乡', '安阳', '洛阳', '信阳','平顶山', '周口', '商丘', '开封', '焦作', '驻马店', '濮阳', '三门峡', '漯河', '许昌', '鹤壁', '济源'],
        '上海':['上海', '松江区', '宝山区', '金山区','嘉定区', '南汇区', '青浦区', '浦东新区', '奉贤区', '闵行区', '徐汇区', '静安区', '黄浦区', '普陀区', '杨浦区', '虹口区', '闸北区', '长宁区', '崇明县', '卢湾'],
        '河北':[ '河北', '石家庄', '唐山', '保定', '邯郸', '邢台',  '沧州', '秦皇岛', '张家口', '衡水', '廊坊', '承德'],
        '浙江':['浙江', '温州', '宁波','杭州', '台州', '嘉兴', '金华', '湖州', '绍兴', '舟山', '丽水', '衢州'],
        '陕西':['陕西', '西安', '咸阳', '宝鸡', '汉中', '渭南','安康', '榆林', '商洛', '延安', '铜川'],
        '湖南':[ '湖南', '长沙', '邵阳', '常德', '衡阳', '株洲', '湘潭', '永州', '岳阳', '怀化', '郴州','娄底', '益阳', '张家界', '湘西土家族苗族自治州'],
        '重庆':[  '重庆', '江北区', '渝北区', '沙坪坝区', '九龙坡区', '万州区', '永川市', '南岸区', '酉阳县', '北碚区', '涪陵区', '秀山县', '巴南区', '渝中区', '石柱县', '忠县', '合川市', '大渡口区', '开县', '长寿区', '荣昌县', '云阳县', '梁平县', '潼南县', '江津市', '彭水县', '璧山县', '綦江县',
         '大足区', '黔江区', '巫溪县', '巫山县', '垫江县', '丰都县', '武隆县', '万盛区', '铜梁区', '南川市', '奉节县', '双桥区', '城口县'],
        '福建':['福建', '漳州', '泉州','厦门', '福州', '莆田', '宁德', '三明', '南平', '龙岩'],
        '天津':['天津', '和平区', '北辰区', '河北区', '河西区', '西青区', '津南区', '东丽区', '武清区','宝坻区', '红桥区', '大港区', '汉沽区', '静海县', '宁河县', '塘沽区', '蓟县', '南开区', '河东区'],
        '云南':[ '云南', '昆明', '红河哈尼族彝族自治州', '大理白族自治州', '文山壮族苗族自治州', '德宏傣族景颇族自治州', '曲靖', '昭通', '楚雄彝族自治州', '保山', '玉溪', '丽江', '临沧', '普洱', '西双版纳傣族自治州', '怒江傈僳族自治州', '迪庆藏族自治州'],
        '四川':['四川省', '成都', '绵阳', '广元','达州', '南充', '德阳', '广安', '阿坝藏族羌族自治州', '巴中', '遂宁', '内江', '凉山彝族自治州', '攀枝花', '乐山', '自贡', '泸州', '雅安', '宜宾', '资阳','眉山', '甘孜藏族自治州'],
        '广西':['广西壮族自治区', '贵港', '玉林', '北海', '南宁', '柳州', '桂林', '梧州', '钦州', '来宾', '河池', '百色', '贺州', '崇左',  '防城港'],
        '安徽':['安徽省', '芜湖', '合肥', '六安', '宿州', '阜阳','安庆', '马鞍山', '蚌埠', '淮北', '淮南', '宣城', '黄山', '铜陵', '亳州','池州', '巢湖', '滁州'],
        '海南':['海南藏族自治州', '三亚', '海口', '琼海', '文昌', '东方', '昌江黎族自治县', '陵水黎族自治县', '乐东黎族自治县', '五指山', '保亭黎族苗族自治县', '澄迈县', '万宁','儋州', '临高县', '白沙黎族自治县', '定安县', '琼中黎族苗族自治县', '屯昌县'],
        '江西':['江西省', '南昌', '赣州', '上饶', '吉安', '九江', '新余', '抚州', '宜春', '景德镇', '萍乡', '鹰潭'],
        '湖北':['湖北省', '武汉', '宜昌', '襄樊', '荆州', '恩施土家族苗族自治州', '孝感', '黄冈', '十堰', '咸宁', '黄石', '仙桃', '随州', '天门', '荆门', '潜江', '鄂州','襄阳'],
        '山西':['山西省', '太原', '大同', '运城', '长治', '晋城', '忻州', '临汾', '吕梁', '晋中', '阳泉', '朔州'],
        '辽宁':['辽宁省', '大连', '沈阳', '丹东', '辽阳', '葫芦岛', '锦州', '朝阳', '营口', '鞍山', '抚顺', '阜新', '本溪',
```

'盘锦', '铁岭'],
'台湾':['台湾省','台北市', '高雄', '台中', '新竹', '基隆', '台南', '嘉义','新北市'],
'黑龙江':['黑龙江', '齐齐哈尔', '哈尔滨', '大庆', '佳木斯', '双鸭山', '牡丹江', '鸡西','黑河', '绥化', '鹤岗', '伊春', '大兴安岭地区', '七台河'],
'内蒙古':['内蒙古自治区', '赤峰', '包头', '通辽', '呼和浩特', '乌海', '鄂尔多斯', '呼伦贝尔','兴安盟', '巴彦淖尔', '乌兰察布', '锡林郭勒盟', '阿拉善盟'],
'香港':["香港","香港特别行政区"],
'澳门':['澳门','澳门特别行政区'],
'贵州':['贵州省', '贵阳', '黔东南苗族侗族自治州', '黔南布依族苗族自治州', '遵义', '黔西南布依族苗族自治州', '毕节', '铜仁','安顺', '六盘水'],
'甘肃':['甘肃省', '兰州', '天水', '庆阳', '武威', '酒泉', '张掖', '陇南', '白银', '定西', '平凉', '嘉峪关', '临夏回族自治州','金昌', '甘南藏族自治州'],
'青海':['青海省', '西宁', '海西蒙古族藏族自治州', '海东', '海北藏族自治州', '玉树藏族自治州', '黄南藏族自治州'],
'新疆':['新疆','新疆维吾尔自治区', '乌鲁木齐', '伊犁哈萨克自治州', '昌吉回族自治州','石河子', '哈密', '阿克苏地区', '巴音郭楞蒙古自治州', '喀什地区', '塔城地区', '克拉玛依', '和田地区', '阿勒泰地区', '吐鲁番', '阿拉尔', '博尔塔拉蒙古自治州', '五家渠',
'克孜勒苏柯尔克孜自治州','北屯','图木舒克'],
'西藏':['西藏区', '拉萨', '山南', '林芝', '日喀则', '阿里地区', '昌都', '那曲'],
'吉林':['吉林省', '吉林市', '长春', '白山', '白城','延边朝鲜族自治州', '松原', '辽源', '通化', '四平'],
'宁夏':['宁夏回族自治区', '银川', '吴忠', '中卫', '石嘴山', '固原']
}
area_list = ['北京','朝阳区', '海淀区', '通州区', '房山区', '丰台区', '昌平区', '大兴区', '顺义区', '西城区', '延庆县', '石景山区', '宣武区', '怀柔区', '崇文区', '密云县',
'东城区', '门头沟区', '平谷区','广东', '东莞', '广州', '中山', '深圳', '惠州', '江门', '珠海', '汕头', '佛山', '湛江', '河源', '肇庆','潮州', '清远', '韶关', '揭阳', '阳江', '云浮', '茂名', '梅州', '汕尾',
'山东', '济南', '青岛', '临沂', '济宁', '菏泽', '烟台','泰安', '淄博', '潍坊', '日照', '威海', '滨州', '东营', '聊城', '德州', '莱芜', '枣庄',
'江苏', '苏州', '徐州', '盐城', '无锡','南京', '南通', '连云港', '常州', '扬州', '镇江', '淮安', '泰州', '宿迁',
'河南', '郑州', '南阳', '新乡', '安阳', '洛阳', '信阳','平顶山', '周口', '商丘', '开封', '焦作', '驻马店', '濮阳', '三门峡', '漯河', '许昌', '鹤壁', '济源',
'上海', '松江区', '宝山区', '金山区','嘉定区', '南汇区', '青浦区', '浦东新区', '奉贤区', '闵行区', '徐汇区', '静安区', '黄浦区', '普陀区', '杨浦区', '虹口区', '闸北区', '长宁区', '崇明县', '卢湾',
'河北', '石家庄', '唐山', '保定', '邯郸', '邢台', '沧州', '秦皇岛', '张家口', '衡水', '廊坊', '承德',
'浙江', '温州', '宁波','杭州', '台州', '嘉兴', '金华', '湖州', '绍兴', '舟山', '丽水', '衢州',
'陕西', '西安', '咸阳', '宝鸡', '汉中', '渭南','安康', '榆林', '商洛', '延安', '铜川',
'湖南', '长沙', '邵阳', '常德', '衡阳', '株洲', '湘潭', '永州', '岳阳', '怀化', '郴州','娄底', '益阳', '张家界', '湘西土家族苗族自治州',
'重庆', '江北区', '渝北区', '沙坪坝区', '九龙坡区', '万州区', '永川市', '南岸区', '酉阳县', '北碚区', '涪陵区', '秀山县', '巴南区', '渝中区', '石柱县', '忠县', '合川市', '大渡口区', '开县', '长寿区', '荣昌县', '云阳县', '梁平县', '潼南县', '江津市', '彭水县', '璧山县', '綦江县',
'大足区', '黔江区', '巫溪县', '巫山县', '垫江县', '丰都县', '武隆县', '万盛区', '铜梁区', '南川市', '奉节县', '双桥区', '城口县',
'福建', '漳州', '泉州','厦门', '福州', '莆田', '宁德', '三明', '南平', '龙岩',
'天津', '和平区', '北辰区', '河北区', '河西区', '西青区', '津南区', '东丽区', '武清区','宝坻区', '红桥区', '大港区', '汉沽区', '静海县', '宁河县', '塘沽区', '蓟县', '南开区', '河东区',

'云南', '昆明', '红河哈尼族彝族自治州', '大理白族自治州', '文山壮族苗族自治州', '德宏傣族景颇族自治州', '曲靖', '昭通', '楚雄彝族自治州', '保山', '玉溪', '丽江', '临沧', '普洱', '西双版纳傣族自治州', '怒江傈僳族自治州', '迪庆藏族自治州',

'四川省', '成都', '绵阳', '广元','达州', '南充', '德阳', '广安', '阿坝藏族羌族自治州', '巴中', '遂宁', '内江', '凉山彝族自治州', '攀枝花', '乐山', '自贡', '泸州', '雅安', '宜宾', '资阳','眉山', '甘孜藏族自治州',

'广西壮族自治区', '贵港', '玉林', '北海', '南宁', '柳州', '桂林', '梧州', '钦州', '来宾', '河池', '百色', '贺州', '崇左', '防城港',

'安徽省', '芜湖', '合肥', '六安', '宿州', '阜阳','安庆', '马鞍山', '蚌埠', '淮北', '淮南', '宣城', '黄山', '铜陵', '亳州','池州', '巢湖', '滁州',

'海南省','海南藏族自治州', '三亚', '海口', '琼海', '文昌', '东方', '昌江黎族自治县', '陵水黎族自治县', '乐东黎族自治县', '五指山', '保亭黎族苗族自治县', '澄迈县', '万宁','儋州', '临高县', '白沙黎族自治县', '定安县', '琼中黎族苗族自治县', '屯昌县',

'江西省', '南昌', '赣州', '上饶', '吉安', '九江', '新余', '抚州', '宜春', '景德镇', '萍乡', '鹰潭',

'湖北省', '武汉', '宜昌', '襄樊', '荆州', '恩施土家族苗族自治州', '孝感', '黄冈', '十堰', '咸宁', '黄石', '仙桃', '随州', '天门', '荆门', '潜江', '鄂州','襄阳',

'山西省', '太原', '大同', '运城', '长治', '晋城', '忻州', '临汾', '吕梁', '晋中', '阳泉', '朔州',

'辽宁省', '大连', '沈阳', '丹东', '辽阳', '葫芦岛', '锦州', '朝阳', '营口', '鞍山', '抚顺', '阜新', '本溪', '盘锦', '铁岭',

'台湾省','台北市', '高雄', '台中', '新竹', '基隆', '台南', '嘉义','新北市',

'黑龙江', '齐齐哈尔', '哈尔滨', '大庆', '佳木斯', '双鸭山', '牡丹江', '鸡西','黑河', '绥化', '鹤岗', '伊春', '大兴安岭地区', '七台河',

'内蒙古自治区', '赤峰', '包头', '通辽', '呼和浩特', '乌海', '鄂尔多斯', '呼伦贝尔','兴安盟', '巴彦淖尔', '乌兰察布', '锡林郭勒盟', '阿拉善盟',

"香港","香港特别行政区",

'澳门','澳门特别行政区',

'贵州省', '贵阳', '黔东南苗族侗族自治州', '黔南布依族苗族自治州', '遵义', '黔西南布依族苗族自治州', '毕节', '铜仁','安顺', '六盘水',

'甘肃省', '兰州', '天水', '庆阳', '武威', '酒泉', '张掖', '陇南', '白银', '定西', '平凉', '嘉峪关', '临夏回族自治州','金昌', '甘南藏族自治州',

'青海省', '西宁', '海西蒙古族藏族自治州', '海东', '海北藏族自治州', '玉树藏族自治州', '黄南藏族自治州',

'新疆','新疆维吾尔自治区', '乌鲁木齐', '伊犁哈萨克自治州', '昌吉回族自治州','石河子', '哈密', '阿克苏地区', '巴音郭楞蒙古自治州', '喀什地区', '塔城地区', '克拉玛依', '和田地区', '阿勒泰地区', '吐鲁番', '阿拉尔', '博尔塔拉蒙古自治州', '五家渠',

'克孜勒苏柯尔克孜自治州', '图木舒克','北屯',

'西藏区', '拉萨', '山南', '林芝', '日喀则', '阿里地区', '昌都', '那曲',

'吉林省', '吉林市', '长春', '白山', '白城','延边朝鲜族自治州', '松原', '辽源', '通化', '四平',

'宁夏回族自治区', '银川', '吴忠', '中卫', '石嘴山', '固原']

```
province = []
for i in user_info1['city_num']:
    if i in area_list:
        for k, v in area_data.items():
            if i in v:
                province.append(k)
    else :
        print(f'需要修改的城市：{i}')
```

```
df1 = pd.DataFrame({'province':province})
df1.to_csv('province.csv',encoding='utf-8')
province = pd.read_csv('province.csv')
user_info1['city_pro'] = province['province']
user_info1
province = user_info1['city_pro'].value_counts().index.tolist()
province_data = user_info1['city_pro'].value_counts().tolist()
df1 = pd.DataFrame({'city':province,'users':province_data})
print(df1)  # 国内各省市数量
map_data = []
for z in zip(province,province_data):
    map_data.append(z)
map=(Map(init_opts=opts.InitOpts(bg_color='#fff'))
    .add(series_name="用户量",data_pair=map_data,maptype="china")
    .set_global_opts(title_opts=opts.TitleOpts("全国各省份用户分布"),
                         visualmap_opts=opts.VisualMapOpts(max_=15000,is_piecewise=True))
    .set_series_opts(label_opts=opts.LabelOpts(is_show=True)))
map.render_notebook()
# 广东省各地区分布情况
gd = user_info1['city_num'][user_info1['city_pro'] == '广东'].value_counts().index.tolist()
gd_data = user_info1['city_num'][user_info1['city_pro'] == '广东'].value_counts().tolist()
pie1 =Pie(init_opts=opts.InitOpts(width='800px'))
pie1.add(
                "",
                [z for z in zip(gd,gd_data)],
                radius = ["38%","80%"],
                center=['65%','55%'],
#                  rosetype="area",
                color='auto')
pie1.set_global_opts(
                title_opts = opts.TitleOpts(title="广东各地区用户量分布"),
                legend_opts = opts.LegendOpts(is_show=False,orient="verticmin",pos_top="15%",pos_left="2%")
)
pie1.set_series_opts(
                label_opts = opts.LabelOpts(font_size=12,formatter="{b}:{c}\n{d}%")
)
pie1.render_notebook()
# 河南省各地区分布情况
hn = user_info1['city_num'][user_info1['city_pro'] == '河南'].value_counts().index.tolist()
hn_data = user_info1['city_num'][user_info1['city_pro'] == '河南'].value_counts().tolist()
pie2 =Pie(init_opts=opts.InitOpts(height='510px',width='800px'))
pie2.add(
                "",
                [z for z in zip(hn,hn_data)],
                radius = ["38%","80%"],
                center=['65%','55%'],
```

```
                rosetype="area",
                color='#999')
pie2.set_global_opts(
                title_opts = opts.TitleOpts(title="河南各地区登录量分布"),
                legend_opts = opts.LegendOpts(orient="verticmin",pos_top="15%",pos_left="2%")
)
pie2.set_series_opts(
                label_opts = opts.LabelOpts(font_size=12,formatter="{b}:{c}\n{d}%")
)
pie2.render_notebook()
city = user_info1['age_month'].groupby(user_info1['city_pro']).mean().index.to_list()
age = user_info1['age_month'].groupby(user_info1['city_pro']).mean().to_list()
df2 = pd.DataFrame({'city':city,'age':age})
# 新建含城市和用户数以及用户年龄数据集 df3
df3 = pd.merge(df1, df2, on='city', how='inner')
print(df3)
# 结合条形图与折线图可视化各省市用户数与年龄分布情况
bar = (
    Bar(init_opts=opts.InitOpts())
    .add_xaxis(df3['city'].tolist())
    .add_yaxis("用户总数",df3['users'].tolist())
    .extend_axis(
                yaxis=opts.AxisOpts(
                    axislabel_opts=opts.LabelOpts(formatter="{value}/月"), interval=5
                )
            )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="国内各省份用户数及年龄分布"))
    .set_series_opts(
        label_opts=opts.LabelOpts(is_show=False),
        markpoint_opts=opts.MarkPointOpts(data=[opts.MarkPointItem(type_="max",name="Max"),

opts.MarkPointItem(type_="min",name="Min")]),
        markline_opts=opts.MarkLineOpts(data=[opts.MarkLineItem(type_="average",name="Average")]))
)
line = (
    Line()
    .add_xaxis(df3['city'].tolist())
    .add_yaxis("平均年龄分布",df3['age'].tolist(), yaxis_index=1)
    .set_series_opts(
                label_opts=opts.LabelOpts(is_show=False),
                markpoint_opts=opts.MarkPointOpts(data=[opts.MarkPointItem(type_="max",name="Max"),

opts.MarkPointItem(type_="min",name="Min")]),

markline_opts=opts.MarkLineOpts(data=[opts.MarkLineItem(type_="average",name="Average")])))
```

```
bar.overlap(line)
bar.render_notebook()
login_day.describe()
login_day.info()
# 按照区间，对用户登录天数的分布情况进行统计分析
login_day_num = pd.cut(login_day['login_day'].values,bins=[-6,-3,0,3,6,9,20,50,200]).value_counts()
login_day_interval = list(login_day_num.index.astype('str'))
print(login_day_num)
# x_data3 = ["[0-30)", "[30-60)", "[60-90)", "[90-120)", "[120-150)", "[150-200)", "[200-300)", '[300-1000)']
# y_data3 = list(temp)
data_pair3 = [list(z) for z in zip(login_day_interval, login_day_num)]
data_pair3.sort(key=lambda x:x[1])
pie3 = (
    Pie(init_opts=opts.InitOpts(width='900px',height='400px',theme='dark',bg_color="#2c343c"))
    .add('登录天数',data_pair3,rosetype='radius',radius='65%',label_opts=opts.LabelOpts(formatter="{b}:{d}%"))
    .set_global_opts(
        title_opts=opts.TitleOpts(title='用户登录天数分布'),
        legend_opts=opts.LegendOpts(is_show=True))
    )
pie3.render_notebook()
# 本小组以 login_time/login_day 表示平均每次登录用户 app 使用时长
num_of_times = login_day['login_time']/login_day['login_day']
num_of_times.describe()
times_num = pd.cut(num_of_times.values,bins=[0,2,4,6,8,10,20,40,60,80,100,500]).value_counts()
times_interval = list(times_num.index.astype('str'))
print(times_num)
# login_day_num = pd.cut(login_day['login_day'].values,bins=[-6,-3,0,3,6,9,20,50,200]).value_counts()
# login_day_interval = list(login_day_num.index.astype('str'))
# print(login_day_num)
line = (Line(init_opts=opts.InitOpts(height='400px',width='800px'))
        .add_xaxis(times_interval)
        .add_yaxis('app 使用时长',times_num)
        .set_global_opts(title_opts=opts.TitleOpts(title='单次用户 app 使用时长'))
        .set_series_opts(label_opts=opts.LabelOpts(),
                         markline_opts=opts.MarkLineOpts(data=[opts.MarkLineItem(type_='max',name="
最佳区间")]))
    )
line.render_notebook()
user_info['datetime'] = pd.to_datetime(user_info['first_order_time'])  # 现将 obj 格式转换为时间格式
user_info['datetime'] = user_info['datetime'].dt.date  # 仅保留日期
total_pv = user_info["user_id"].count()
print("这个时间段内 app 的总访量为：{}".format(total_pv))
total_uv = user_info["user_id"].nunique()
print("这个时间段内独立访客数量为：{}".format(total_uv))
date_pv = user_info.groupby('datetime')["user_id"].count()
```

```
print(date_pv)
date_uv = user_info.groupby('datetime')["user_id"].apply(lambda x:x.nunique())
print(date_uv)
plt.figure(figsize=(16,10))
plt.subplot(2,1,1)
plt.plot(date_pv,c="r")
plt.title('PV')
plt.grid(color='y', linestyle='--', linewidth=0.5)
plt.subplot(2,1,2)
plt.plot(date_pv,c="g")
plt.title('UV')
plt.tight_layout()
plt.grid(color='r', linestyle='--', linewidth=0.5)
plt.show()
```

**#任务 3：构建模型判断用户最终是否会下单购买或下单购买的概率，并将模型结果输出为 csv 文件**
**#决策树代码（R 软件）**

```
#载入包
install.packages('lubridate')
iinstall.packages('pROC')
iinstall.packages('rpart.plot')
library(lubridate)
library(pROC)
library(rpart.plot)
#预处理数据
options(digits=20)#将数字精度设置为 20 位
user_info=read.csv('user_info.csv',sep=',',header=TRUE,encoding='UTF-8')
login_day=read.csv('login_day.csv',sep=',',header=TRUE,encoding='UTF-8')
visit_info=read.csv('visit_info.csv',sep=',',header=TRUE,encoding='UTF-8')
result = read.csv('result.csv',sep=',',header=TRUE,encoding='UTF-8')
sum(result)
colnames(result)[1]='user_id'
colnames(user_info)[1]='user_id'
colnames(visit_info)[1]='user_id'
user_info=user_info[,c(1,2,3,4,6,7,8)]   #删除 city_num 的列
user_info=user_info[user_info$age_month<=144,]   # 年龄限制在 12 周岁内
user_info1=merge(user_info,login_day,by=c('user_id'))
user_info2=merge(user_info1,visit_info,by=c('user_id'))
user_info2$first_order_time=year(as.Date(user_info2$first_order_time))#提取年份
user_info2$result=0
for (i in 1:nrow(user_info2)){
   if (user_info2$user_id[i] %in% result$user_id){
      user_info2$result[i]=1
   }
}
sum(user_info2$result)
```

```
alldata = user_info2
alldata$result = as.factor(alldata$result)
#过采样
hang = which(alldata$result == 1,)
set.seed(1234)
hang1 = sample(hang,120000,replace = TRUE)
newdata = alldata[hang1,]
alldata1 = rbind(alldata,newdata)
nrow(alldata1)
nrow(newdata)
id = alldata[,1]
alldata1 = alldata1[,-1]
#随机生成训练集（70%）和测试集（30%）
set.seed(1234)
traingroupid = sample(nrow(alldata1),ceiling(nrow(alldata1)*0.7),replace = F)
w = alldata1[traingroupid,]#训练集
testgroup = alldata1[-traingroupid,]#测试集
nrow(w)
nrow(testgroup)
#构建决策树
b=rpart(result~.,w)
b
par(mfrow = c(1,1))
rpart.plot(b,extra = 1,type = 2)
a =predict(b,w,type = 'class')
a1 = predict(b,testgroup,type = 'class')
predict1 = predict(b,alldata,type = 'class')
table(w$result,a)#训练集混淆矩阵
table(testgroup$result,a1)#测试集混淆矩阵
mean(w$result!= a)#训练集误判率
mean(testgroup$result!= a1)#测试集误判率
mean(predict1!=alldata$result)#基于有效样本误判率
table(alldata$result,predict1)#基于有效样本混淆矩阵
#查看 cp 值
printcp(b)
#画 ROC 曲线
a1 = as.numeric(a1)
testgroup$result = as.numeric(testgroup$result)
modelroc <- roc(a1,testgroup$result)
plot(modelroc, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
plot(modelroc,print.auc=TRUE,auc.polygon=TRUE,
     grid=c(0.1,0.2),grid.col=c("green","red"),
     max.auc.polygon=TRUE,auc.polygon.col="skyblue",print.thres=TRUE)
#保存预测结果
```

```
predict1 = as.matrix(as.numeric(predict1)-1,nrow(alldata),1)
options(digits=20)
sample_out = cbind(id,predict1)
colnames(sample_out) = c('user_id','result')
head(sample_out)
write.csv(as.matrix(sample_out),'sample_output.csv',quote = FALSE,row.names=F)
```

**#任务 4：用户消费行为价值分析**

```
login_users_counts = user_info[user_info['app_num']!=0].shape[0]
join_course_users_counts = login_day[login_day['camp_num']!=0].shape[0]
start_study_users_counts = login_day[login_day['learn_num']!=0].shape[0]
finish_num=list(login_day.groupby('finish_num')['user_id'].count().index)
user_counts=list(login_day.groupby('finish_num')['user_id'].count().values)
plt.plot(finish_num,user_counts,color='r',label='linear',marker='o')
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.xlabel("完成课程数")
plt.ylabel("用户数")
plt.title("完成课程人数情况")
finish_5courses_users_counts = login_day[login_day['finish_num']>=5].shape[0]
#计算总体转化率和环节转化率
attr = ["注册","加入课程","开始学习","结课"]
values = [login_users_counts,join_course_users_counts,start_study_users_counts,finish_5courses_users_counts]
attr_data = pd.DataFrame({"环节":attr,"人数":values})
attr_data["总体转化率"] = round(attr_data.人数*100/attr_data.iloc[0,1],2)
attr_data["环节转化率"] = round(attr_data.人数*100/attr_data.人数.shift(1).fillna(133935),2)
print(attr_data)
#漏斗图-总体转化率
funnel1= Funnel(init_opts=opts.InitOpts(bg_color='#d9d6c3'))
funnel1.add(
      is_selected = True,
      series_name = "总体转化率",
      data_pair = [ z for z in zip(attr_data.环节,attr_data.总体转化率)],
                        )
funnel1.set_series_opts(
            label_opts = opts.LabelOpts(formatter='{b}:{c}%'),
)
funnel1.render_notebook()
#  漏斗图-单一环节转化率
funnel2= Funnel(init_opts=opts.InitOpts(bg_color='#d9d6c3'))
funnel2.add(
      is_selected = True,
      series_name = "环节转化率",
      data_pair = [ z for z in zip(attr_data.环节,attr_data.环节转化率)],
                        )
funnel2.set_series_opts(
```

```
        label_opts = opts.LabelOpts(formatter='{b}:{c}%'),
)
funnel2.render_notebook()
#新建 consumer_behavior 数据集包含变量 user_id、first_order_time、first_order_price、coupon、coupon_visit、click_buy、click_notUnlocked,应用于用户消费者行为价值分析
consumer_behavior = pd.DataFrame({'user_id':visit_info['user_id'],

'coupon_visit':visit_info['coupon_visit'],'click_buy':visit_info['click_buy'],
                                  'click_notunlocked':visit_info['click_notunlocked']
                                 })
consumer_behavior=pd.merge(consumer_behavior,user_info[['user_id','first_order_time','first_order_price']],on='user_id')
consumer_behavior = pd.merge(consumer_behavior,login_day[['user_id','coupon']],on='user_id')
# 用户购买行为与领券行为的相关性分析
consumer_behavior1 = consumer_behavior.drop(columns=['first_order_time'])
print(consumer_behavior1)
consumer_behavior2 = consumer_behavior1.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
#画出体验课下单时间的人数分布图
plt.plot(user_info.groupby('datetime')['user_id'].count().index,user_info.groupby('datetime')['user_id'].count().values,color='r')
plt.xlabel('first_order_time')
plt.ylabel('user_counts')
#计算 R、M 值
user=user_info
user['datetime'] = pd.to_datetime(user['first_order_time'])
user['year']=user['datetime'].dt.year
user['month']=user['datetime'].dt.month
user['day']=user['datetime'].dt.day
user_value_id=user_info[(user['year']==2018)&(user['month']==12)]
user_value_id.index=range(0,len(user_value_id))
user_value=pd.DataFrame((np.ones((len(user_value_id),8))))
user_value.columns=('user_id','R','M','R 值打分','M 值打分','R 值高低','M 值高低','用户分类')
user_value['user_id']=user_value_id['user_id']
user_value['R']=31-user_value_id['day']
user_value['M']=user_value_id['first_order_price']
print(user_value['R'].describe())
print(user_value['M'].describe())
#给 R、M 值按照指标的价值打分
user_value.loc[user_value['R']<3,'R 值打分']=5
user_value.loc[(user_value['R']>=3)&(user_value['R']<5),'R 值打分']=4
user_value.loc[(user_value['R']>=5)&(user_value['R']<10),'R 值打分']=3
user_value.loc[(user_value['R']>=10)&(user_value['R']<20),'R 值打分']=2
user_value.loc[user_value['R']>=20,'R 值打分']=1
user_value.loc[user_value['M']<10,'M 值打分']=1
user_value.loc[(user_value['M']>=10)&(user_value['M']<50),'M 值打分']=2
user_value.loc[(user_value['M']>=50)&(user_value['M']<90),'M 值打分']=3
```

```
user_value.loc[(user_value['M']>=90)&(user_value['M']<120),'M 值打分']=4
user_value.loc[user_value['M']>=120,'M 值打分']=5
#计算 R、M 值高低
user_value.loc[user_value['R 值打分']>user_value['R 值打分'].mean(),'R 值高低']='高'
user_value.loc[user_value['R 值打分']<=user_value['R 值打分'].mean(),'R 值高低']='低'
user_value.loc[user_value['M 值打分']>user_value['M 值打分'].mean(),'M 值高低']='高'
user_value.loc[user_value['M 值打分']<=user_value['M 值打分'].mean(),'M 值高低']='低'
#用户分类
user_value.loc[(user_value['R 值高低']=='高')&(user_value['M 值高低']=='高'),'用户分类']='重要发展用户'
user_value.loc[(user_value['R 值高低']=='低')&(user_value['M 值高低']=='高'),'用户分类']='重要挽留用户'
user_value.loc[(user_value['R 值高低']=='高')&(user_value['M 值高低']=='低'),'用户分类']='一般发展用户'
user_value.loc[(user_value['R 值高低']=='低')&(user_value['M 值高低']=='低'),'用户分类']='一般挽留用户'
user_value.groupby('用户分类')['user_id'].count()
```