# 10.1　智能手机用户监测数据案例分析

## 10.1.1　数据简介

该数据来自 QuestMobile 公司(www.questmobile.cn)某年连续 30 天的 4 万多智能手机用户的监测数据,已经做了脱敏和数据变换的处理。每天的数据为 1 个 txt 文件,共 10 列,记录了每个用户(以 uid 为唯一标识)每天使用各款 APP(以 appid 为唯一标识)的起始时间、使用时长、上下行流量等。具体说明见表 10-1。此外,有一个辅助表格,app_class.csv,共两列。第一列是 appid,给出 4000 多个常用 APP 所属类别(app_class),比如视频类、游戏类、社交类等,用英文字母 a~t 表示。其余 APP 不常用,所属类别未知。该数据可以在人大出版社提供的网址下载。本案例的所有程序也可以在人大出版社提供的网址下载。

表 10-1 智能手机用户的监测数据说明

| 变量编号 | 变量名 | 释义 |
|---|---|---|
| 1 | uid | 用户的 id |
| 2 | appid | APP 的 id (与 app_class 文件中的第一列对应) |
| 3 | app_type | APP 类型:系统自带、用户安装 |
| 4 | start_day | 使用起始天,取值 1~30(注:第 1 天数据的头两行的使用起始天取值为 0,说明是在这一天的前一天开始使用的) |
| 5 | start_time | 使用起始时间 |
| 6 | end_day | 使用结束天 |
| 7 | end_time | 使用结束时间 |
| 8 | duration | 使用时长(秒) |
| 9 | up_flow | 上行流量 |
| 10 | down_flow | 下行流量 |

## 10.1.2　单机实现

### 1.描述统计分析

(1)用户记录的有效情况。互联网数据的记录过程比较复杂,总是会由于各种各样的原因使得用户的记录存在缺失的情况,读者在进行实际项目分析时,一定要具体情况具体分析。对于本案例,如果一个用户在一整天中没有任何一条 APP 使用记录,则该用户在该天记录缺失。依据这个原则可以统计每位用户在 30 天中的有效记录天数,图 10-1 展示了用户缺失天数(即 30-有效天数)的频数分布直方图。

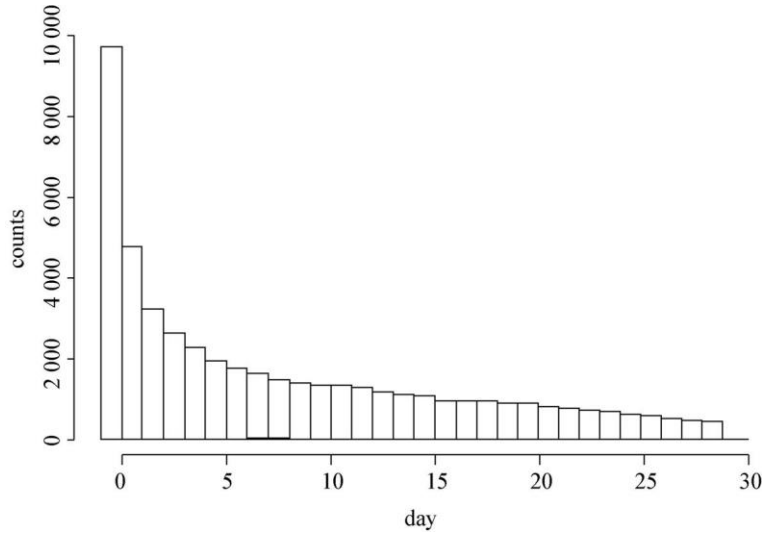


图 10-1　用户缺失天数频数分布直方图

可以看到在全部 48179 名用户当中,9700 名用户不存在缺失,5 天之后缺失的人数开始接近线性递减,缺失天数小于 5 的用户为 22614 人,接近用户总体的一半。缺失天数小于 10 的用户为 30767 人。可以看出不同用户缺失天数存在较大差异,在后面的分析中会考虑这个因素。

(2)各类 APP 的使用强度和相关性。接下来统计一下各类 APP 的用户有效使用强度和相关性。通过编写 Python 程序实现这一目的,程序中需要实现的内容包括:

1)对每天的每条数据记录,通过 appid 与 app_class.csv 文件中的 app_class 对应。

2)对每一天的数据,根据 uid 及 app_class 两个字段进行分类汇总,得到每人每天使用每种类别 APP 的总时长。

3)汇总 30 天的数据,得到每人使用每种类别 APP 的总时长(有效观测天数内的总时长)。

由于每类 APP 的内容不同、涵盖范围不同、面向的人群不同,导致不同种类的 APP 的使用情况不同。表 10-2 展示了各类 APP 的使用强度(有效观测天数的日均使用时长,因为数据取值有 0 且高度右偏,所以数据加 1 之后取对数)。

表 10-2 各类 APP 使用强度(对数变换)单位:秒

| 编号 | APP 类型 | 均值 | 标准差 | 最小值 | 最大值 |
|---|---|---|---|---|---|
| 1 | a | 2.51 | 2.63 | 0.00 | 16.62 |
| 2 | b | 1.47 | 2.06 | 0.00 | 17.90 |
| 3 | c | 4.40 | 2.87 | 0.00 | 17.06 |
| 4 | d | 4.32 | 2.24 | 0.00 | 17.83 |
| 5 | e | 2.82 | 3.04 | 0.00 | 17.74 |
| 6 | f | 6.96 | 1.97 | 0.00 | 18.64 |
| 7 | g | 5.16 | 2.17 | 0.00 | 19.07 |
| 8 | h | 0.90 | 2.15 | 0.00 | 19.47 |
| 9 | i | 2.14 | 2.29 | 0.00 | 18.78 |
| 10 | j | 0.99 | 2.23 | 0.00 | 16.43 |
| 11 | k | 2.34 | 2.33 | 0.00 | 19.88 |

| | | | | | |
|---|---|---|---|---|---|
| 12 | l | 0.17 | 0.78 | 0.00 | 15.04 |
| 13 | m | 0.20 | 0.93 | 0.00 | 8.86 |
| 14 | n | 1.49 | 1.77 | 0.00 | 16.62 |
| 15 | o | 0.90 | 1.87 | 0.00 | 11.93 |
| 16 | p | 2.02 | 2.53 | 0.00 | 17.68 |
| 17 | q | 1.96 | 2.20 | 0.00 | 18.37 |
| 18 | r | 0.08 | 0.53 | 0.00 | 7.40 |
| 19 | s | 0.60 | 1.39 | 0.00 | 9.42 |
| 20 | t | 3.56 | 3.24 | 0.00 | 18.40 |

由表 10-2 可知,f,g,c,d,t 的日有效使用时长按先后顺序在所有 APP 类型中排名前五,相比其他 APP 使用强度较大。

利用 48179 位用户对 20 类 APP 的有效日均使用时长可以计算出任意两类 APP 应用之间的相关系数。以相关系数大小为面积,可通过图 10-2 反映各类 APP 之间的线性相关程度。

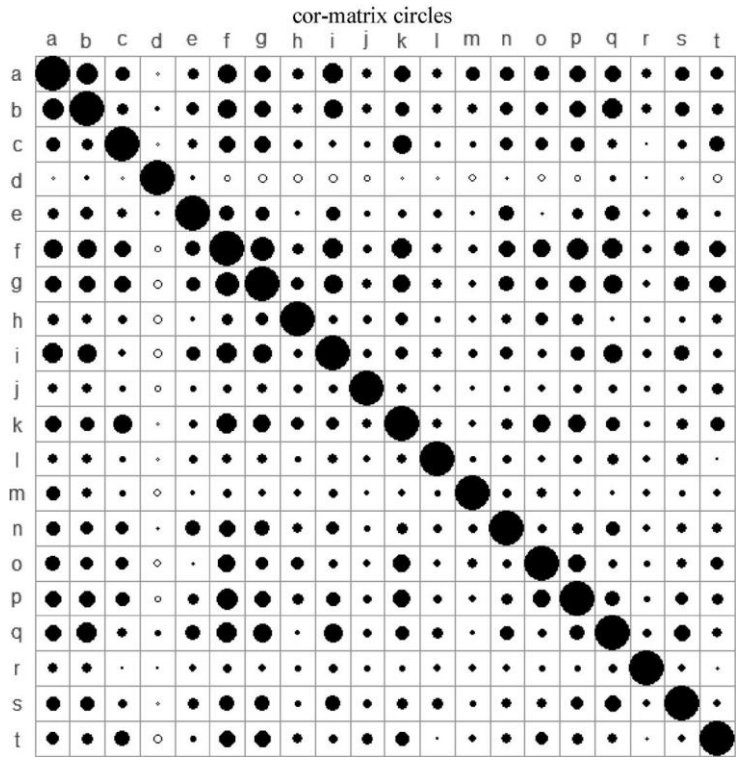


图 10-2　各类 APP 间的相关系数

在图 10-2 中,黑点的面积大小表示两类 APP 相关性的强弱。由图可知,总体上,各类 APP 之间的线性关系不是很显著,部分 APP 之间存在一定的相关性,如 a 类 APP 与 b,c,f,g,i,k,o,p,q 类 APP 的相关性较强,f 类 APP 与 a,b,c,g,i,k,o,p,q 类 APP 存在较强的相关性。因此,用户可能会同时使用不同种类的 APP,而且不同的 APP 间可能存在一定的关联,成为较为固定的“搭配”。

在此,我们只给出最基础的描述分析。请读者自行尝试其他更多情况下的描述统计分析,以对数据有进一步的了解。

## 2. APP 使用情况预测分析

本节对用户使用 APP 的情况进行预测。我们要研究的问题是通过用户的 APP 使用记录预测用户未来是否使用 APP(分类问题)及使用时长(回归问题)。

(1)分类。我们根据用户第 1~23 天的某类 APP 的使用情况,来预测用户在第 24~30 天是否会使用该类 APP。这里以 i 类为例。因变量 $y$ 为二分类变量,$y$=1 表示用户在第 24~30 天使用了该类 APP,$y$ =0 表示没有使用该类 APP。用于预测的变量说明如表 10-3 所示。

表 10-3 因变量和自变量说明

| 变量符号 | 变量名称的含义 | 类型 | 单位/说明 |
|---|---|---|---|
| $y$ | 第 24~30 天是否使用该类 APP | 分类变量 | $y$=1 使用;<br>$y$=0 未使用 |
| $x1$ | 第 24 天前最后一次使用该类 APP 的日期距离第 24 天的天数(市场营销领域中的“近度”(Recency)变量) | 连续变量 | 天 |
| $x2$ | 第 24 天前最后一次使用那天的使用强度 | 连续变量 | 秒 |
| $x3$ | 前 23 天使用总天数除以有效观测天数(市场营销领域中的“频度”(Frequency)变量) | 连续变量 | 无 |
| $x4$ | 前 23 天使用天数当中的平均使用强度(市场营销领域中的“强度”(Monetary)变量) | 连续变量 | 秒 |
| $x5$ | 前 23 天有效观测天数当中的平均使用强度 | 连续变量 | 秒 |
| $x6$ | 第 24 天前一天(第 23 天)的使用强度 | 连续变量 | 秒 |
| $x7$ | 第 24 天前一周内(第 17~23 天)的有效观测天数当中的平均使用强度 | 连续变量 | 秒 |
| $x8$ | 第 24 天前一周外(第 1~16 天)的有效观测天数当中的平均使用强度 | 连续变量 | 秒 |

在预测前,首先删除前 23 天当中不存在有效观测天数的用户,共 571 名。之后,删除后 7 天当中不存在有效观测天数的用户,共 3223 名。最后保留的用户个数为 44385。$x1$ ~$x2$ 分别存在 27442 个缺失,占总用户数的 50.56%。$x6$ ~$x8$ 也存在缺失的情形,缺失数据分别占总用户数的 19.78%,3.7%,1.6%。对自变量中的缺失值使用中位数插补法,读者也可以尝试其他缺失数据处理方法。此外,读者也可以考虑针对后 7 天数据中存在缺失数据的更好的处理方法。

统计可知,$y$=0 为 23241 个(52.36%),$y$=1 为 21144 个(47.64%)。随机选取 80%作为训练集,20%作为测试集,模型选用随机森林。利用 $x1$ ~$x8$ 指标对 $y$ 指标建立分类模型。对 i 类 APP 的预测结果如表 10-4 所示,整体准确率为 81.77 %,变量重要性见图 10-3。

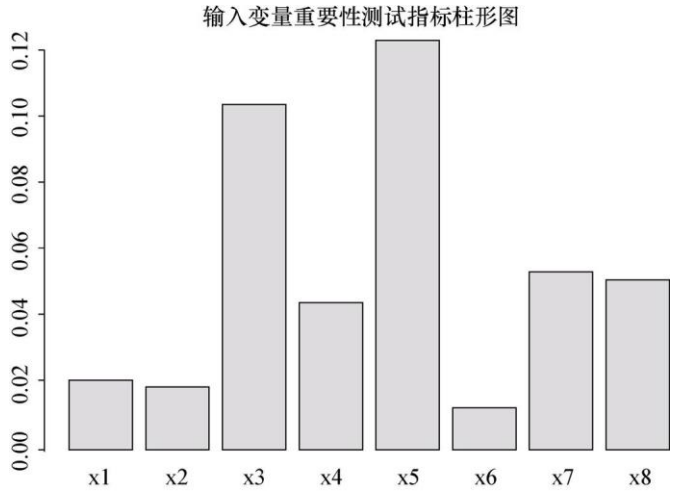


图 10-3　随机森林变量重要性(分类)

表 10-4 随机森林测试集混淆矩阵

| Predict | True | |
|---|---|---|
| | 0 | 1 |
| 0 | 3904 | 882 |
| 1 | 715 | 3259 |

读者可以尝试更多的方法,构建更多有意义的自变量并进行分析;也可以对某个 APP 而不是此处的 APP 类别进行分析。

(2)回归。与上一部分分类不同的是,这里要预测的是第 24~30 天用户使用某类 APP 的有效日均使用时长,因而因变量是连续变量,自变量的选取不变,我们要预测的 APP 是 i 类。

这里预测模型选取的是随机森林。与分类预测不同的是,比较各个模型的标准是 NMSE:

$$NMSE=\sqrt{\frac{\sum\left(y_i-\hat{y}_i\right)^2}{\sum(y_i-\bar{y})^2}}$$

式中,$y_i$表示使用时长实际值;$\hat{y}_i$表示使用时长预测值;$\bar{y}$表示所有用户的实际使用时长的平均值。不使用任何模型时,我们可以用数据的平均值作为预测值,此时的预测误差可表示为$\sum(y_i-\bar{y})^2$;当使用模型预测时,模型的预测误差可表示为$\sum(y_i-\hat{y}_i)^2$。若$\sum(y_i-\hat{y}_i)^2$比$\sum(y_i-\bar{y})^2$小,表示用模型做预测是有意义的,否则表示用$\bar{y}$做预测效果反而更好。*NMSE* 的取值越小表示模型的预测效果越好。

运行随机森林模型(因变量加 1 之后取对数)得到的结果是,测试集的 *NMSE* 为 0.621。变量重要性见图 10-4。

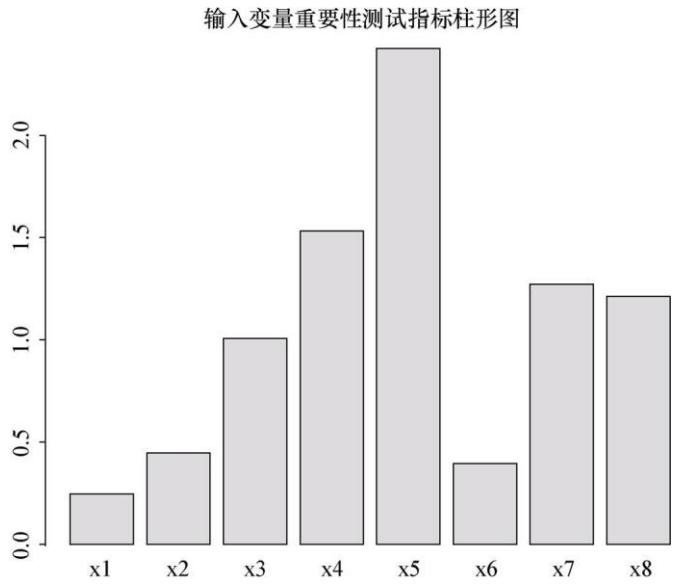


图 10-4　随机森林变量重要性(回归)

3.用户行为聚类分析

(1)用户 APP 使用差异情况聚类。对于在描述统计分析中得到的用户对 20 类 APP 有效使用天数的日均使用强度数据(对数变换之后),我们选用 K 均值聚类。由于聚类结果和初始赋值有关,因此在 R 程序中,我们随机重复 20 次初始赋值。评价聚类效果的常用标准是组间方差占总方差的比重。若该比重较大,说明各类别组内方差较小,类内同质化程度高,因而聚类效果较好。若该比重较小,则表明聚类效果较差。

在 K 均值聚类方法中,需要事先设定待分类的个数 *K*。这里我们将 *K*的取值从 2 到 15 依次运行,根据分类结果选择合理的 *K* 值。根据 K 均值聚类方法,聚类结果如表 10-5 所示。

表 10-5 不同 *K*值下组间方差与总方差的占比

| *K*值 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 组间方差/总方差 | 0.13 | 0.19 | 0.23 | 0.26 | 0.28 | 0.30 | 0.32 |
| | | | | | | | |
| *K*值 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 组间方差/总方差 | 0.33 | 0.35 | 0.36 | 0.37 | 0.38 | 0.39 | 0.39 |

由表 10-5 可知,当 *K*值从 2 增加到 5 时,组间方差的占比上升较快,每增加一个类,组间方差/总方差上升 4%左右;*K* 值在 5 之后上升的幅度趋于稳定,在大于等于 8 之后保持在 2%及以下的水平。因此,我们认为 *K*取 8,即将用户分为 8 个类较为合适。当 *K*=8 时,组间方差/总方差为 0.32,相应的各类中心点的值如表 10-6 所示。

表 10-6 各类的中心值

| 类别/APP | a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.71 | 0.39 | 1.50 | 4.39 | 0.90 | 4.71 | 3.04 | 0.40 | 0.78 | 0.48 |
| 2 | 3.69 | 2.13 | 5.50 | 3.83 | 2.85 | 7.77 | 6.07 | 7.16 | 2.75 | 1.23 |
| 3 | 5.37 | 3.59 | 6.03 | 4.32 | 4.15 | 8.15 | 6.32 | 0.57 | 3.98 | 0.80 |
| 4 | 1.88 | 1.11 | 4.85 | 4.46 | 6.12 | 7.23 | 5.25 | 0.39 | 1.82 | 0.35 |
| 5 | 1.89 | 0.96 | 5.08 | 4.39 | 0.26 | 7.09 | 5.38 | 0.35 | 1.53 | 0.23 |
| 6 | 3.12 | 1.85 | 4.84 | 4.08 | 2.68 | 7.31 | 5.79 | 0.74 | 2.53 | 7.01 |
| 7 | 2.24 | 1.35 | 4.04 | 4.46 | 6.25 | 7.23 | 5.18 | 0.37 | 2.40 | 0.53 |
| 8 | 2.78 | 1.40 | 4.78 | 4.36 | 0.33 | 7.42 | 5.63 | 0.36 | 2.41 | 0.37 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 类别/APP | k | l | m | n | o | p | q | r | s | t |
| 1 | 0.64 | 0.05 | 0.08 | 0.66 | 0.24 | 0.62 | 0.57 | 0.03 | 0.16 | 1.10 |
| 2 | 3.62 | 0.27 | 0.35 | 1.97 | 1.89 | 3.19 | 2.16 | 0.16 | 0.79 | 4.79 |
| 3 | 4.02 | 0.35 | 0.42 | 2.14 | 2.17 | 4.54 | 3.67 | 0.18 | 1.25 | 5.40 |
| 4 | 2.11 | 0.14 | 0.16 | 1.66 | 0.60 | 1.48 | 1.90 | 0.08 | 0.50 | 6.47 |
| 5 | 2.43 | 0.11 | 0.19 | 1.28 | 0.82 | 1.62 | 1.53 | 0.04 | 0.41 | 6.40 |
| 6 | 2.88 | 0.21 | 0.23 | 1.65 | 1.08 | 2.29 | 2.35 | 0.10 | 0.80 | 5.30 |
| 7 | 1.84 | 0.18 | 0.16 | 1.77 | 0.48 | 1.69 | 2.24 | 0.10 | 0.66 | 0.32 |
| 8 | 2.58 | 0.16 | 0.17 | 1.40 | 0.77 | 2.03 | 2.05 | 0.06 | 0.60 | 0.44 |

由表 10-6 可知,这 8 类用户都特别喜欢 f 类 APP,除此之外的突出特点是:第 1 类(7818 人,约占 16.2%)用户对各类 APP 整体使用较少,其中对 d 类 APP 使用相对频繁;第 2 类用户(3332 人,约占 6.9%)非常喜欢使用 h 类 APP;第 3 类用户(6001 人,约占 12.5%)非常喜欢 g 类 APP;第 4 类用户(6083 人,约占 12.6%)非常喜欢使用 t 类 APP;第 5 类用户(8009 人,约占 16.6%)非常喜欢使用 t 类 APP;第 6 类用户(3600 人,约占 7.5%)喜欢 j 类 APP;第 7 类用户(6924 人,约占 14.4%)非常喜欢使用 e 类 APP;第 8 类用户(6412 人,约占 13.3%)非常喜欢使用 g 类 APP。

(2)双向聚类。在上文所用的 K 均值聚类中,每个类别中心(即均值)的计算使用了所有类别 APP(即所有列)的信息,最终将所有用户(即所有行)划分到 *K* 个类别中。这种聚类的特点是使用了所有行列的信息,但有时候这种聚类方式的效果不一定理想。在本案例中,绝大部分用户仅使用 20 类 APP 中的部分 APP,因此用户在部分属性而非全部属性上表现出相似性。此外,在传统聚类方法中,用户会被划分到某一类中,但实际上用户可能具备多个类别的特征。在一些实际问题中,如果仅用部分行列的信息进行聚类,可能会得到更灵活的聚类结果,并且可以克服上述缺点,这就是第 8 章介绍的双向聚类。本节使用双向聚类对用户特征进行分析。

在本案例中,我们使用这种算法来分析用户对 APP 的使用情况。我们将用户对某类 APP 的使用情况分为经常使用和不经常使用两种情况。如果用户对该类 APP 的使用时长大于全部用户对该类 APP 有效日均使用时长的中位数,则认为此用户经常使用该 APP,取值为 1,否则取值为 0,认为此用户不经常使用该 APP。将用户 APP 使用时长矩阵转换为二值矩阵,在该矩阵的基础上,在 R 中用 biclust 包中的 BCBimax()函数实现双向聚类。为了使聚类结果更有代表性,在聚类时规定每类结果的用户人数不少于 7500,且每类用户所用的 APP 不少于 4 类,双向聚类结果如表 10-7 所示。

表 10-7 第一次双向聚类结果

| 类别 | 相关 APP class | | | | 人数 |
|---|---|---|---|---|---|
| 1 | f | g | i | q | 7200 |
| 2 | a | b | i | q | 7560 |
| 3 | a | f | i | q | 7489 |

由表 10-7 可以看出,每类结果都包含 4 个 APP 类型,聚类结果共涉及 6 类 APP。鉴于双向聚类的行列非排他性,各大类别的用户会出现重叠现象,说明进入第一次双向聚类结果的用户总体不是表 10-7 人数列的简单加和,进一步计算表明进入第一次聚类的用户共有 11293 人,其中 4065 人同时使用 6 类 APP。

在 biclust 包中,可以使用 heatmapBC()函数画出热力图,对原矩阵的行列做出适当调整后观察聚类效果,如图 10-5 所示。

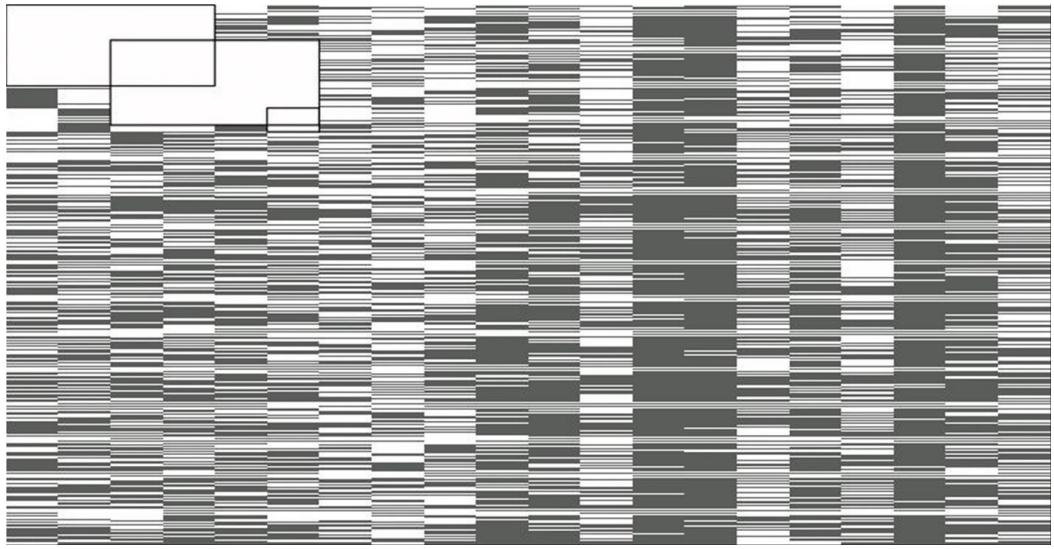

图 10-5　第一次双向聚类热力图

在图 10-5 中,我们可以看到左上角的三个长方形就是对应的双向聚类结果,用户使用 APP 的共同特征集中在变换后的矩阵的左上角。

为了发现更多用户的特征,可以对双向聚类后余下的用户进行第二次双向聚类。在第二次双向聚类中,我们规定每类结果的用户人数不少于 5600,且每类用户所用的 APP 不少于 3 类。第二次双向聚类结果如表 10-8 所示。

表 10-8 第二次双向聚类结果

| 类别 | 相关 APP class | | | 人数 |
|---|---|---|---|---|
| 1 | c | k | t | 5846 |
| 2 | c | k | p | 5322 |

由上表可以看出,每类结果都与 3 个类型相关,聚类结果共涉及 4 类 APP,同理还是考虑双向聚类的行列非排他性,进一步从 APP 类型出发对双向聚类结果进行探讨(见表 10-9)。

表 10-9 第二次双向聚类结果交叉分析

| 类别 | 相关 APP class | | | | 人数 |
|---|---|---|---|---|---|
| 同时使用 4 个 APP | c | k | p | t | 3641 |
| 只使用第 1 类 | c | k | p | — | 2030 |
| 只使用第 2 类 | c | k | — | t | 2587 |

在第二次双向聚类中,经统计共有 8258 名用户进入聚类结果,进入聚类结果的共有 c,k,p,t 四类 APP;其中 3641 名用户经常使用 c,k,p,t 这四类 APP,2030 名用户经常使用 c,k,p 这三类 APP(对应聚类结果第 1 类),2587 名用户经常使用 c,k,t 这三类 APP(对应聚类结果第 2 类)。相应的热力图如图 10-6 所示。

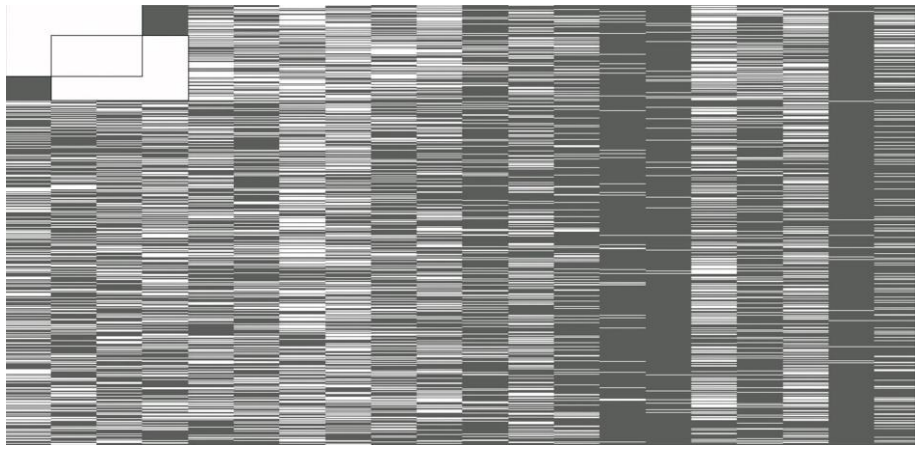

图 10-6　第二次双向聚类热力图

(3)RFM 聚类。在现实的业务分析过程中,相对于比较用户对不同类别 APP 的总的使用强度而言,人们更加关注用户对某一种特定 APP 的使用行为。用户对特定 APP 的使用行为包含开始使用、保持使用以及流失的行为和过程等。针对这种需求,可以利用每个人对某一种特定 APP 在连续 30 天中的使用情况作为原始数据对人进行聚类,从而区分出对一款 APP 具有不同使用行为特征的人群。我们以编号为 17442 的 APP 为例,考察在观测的 30 天内使用过该 APP 且有效观测天数大于等于 26 天的 8718 位用户的行为特征。基于原始数据,借鉴度量消费者行为的三个重要指标 RFM——最近一次消费(Recency)、消费频率(Frequency)和消费金额(Monetary),针对 APP 数据构造最近一次使用(最近一次使用距离最后一天的天数)、使用频率(使用天数除以有效观测天数)和有效使用时长(使用总时长除以使用天数)三个指标,以标准化后的这三个变量作为特征对人群进行聚类分析。

在计算 Recency 指标时,会遇到缺失数据的情况,经统计在 8718 个样本中有 520 个观测出现该现象,占总样本的 5.9%,我们将这部分数据删除,保留 8198 个样本进行下一步分析。

根据组间方差的占比随类别的变化情况将人群分为 4 类,组间方差占总方差的 76.1%,聚类效果较好,各类别的类中心见表 10-10 和图 10-7。

表 10-10RFM 聚类各类中心

| | 第 1 类 | 第 2 类 | 第 3 类 | 第 4 类 |
|---|---|---|---|---|
| R | -0.53708 | -0.50682 | -0.12983 | 2.367031 |
| F | 1.301098 | 0.874899 | -0.65794 | -1.18991 |
| M | 2.206248 | 0.122648 | -0.48931 | -0.57305 |
| 各类人数 | 905 | 2703 | 3608 | 982 |

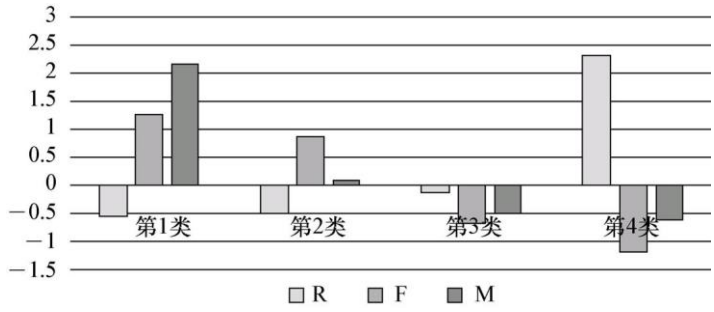


图 10-7　RFM 聚类各类中心

由各类中心可以看出,第 1~4 类用户的使用频率(F)越来越低,使用时长(M)越来越短,最后一次使用时间距离最后一次观测(R)越来越远。由此可以看出第 1 类用户为该 APP 的忠诚使用者,第 2 类用户为频繁使用者,第 3 类为轻度使用者,第 4 类为流失使用者。从公司运营层面考虑,对于不同的用户可以采取不同的策略,以获取更多利润。

最后给出各类用户使用强度的热力图(见图 10-8)。这是在 Excel 中完成的,具体做法如下:首先在每类用户中随机选取 200 名用户,每一类用户共 30 列,代表 1~30 天,表格中的数字代表该用户在这一天的使用时长,空白为 0,表示没有使用,用白色填充。大于 0 的部分,分段处理,颜色依次加深。需要说明的是,对于缺失数据,我们用浅绿色进行处理,在黑白印刷的书上很难呈现。可以看到四类人群在使用行为上有差距。第一类用户经常使用该 APP,且每次使用强度非常大;第二类用户同样经常使用,但是强度不如第一类用户;第三类用户不经常使用该 APP 或者刚刚开始使用该 APP;第四类用户偶尔使用该 APP 但是现在已经超过 10 天没有使用过该 APP。

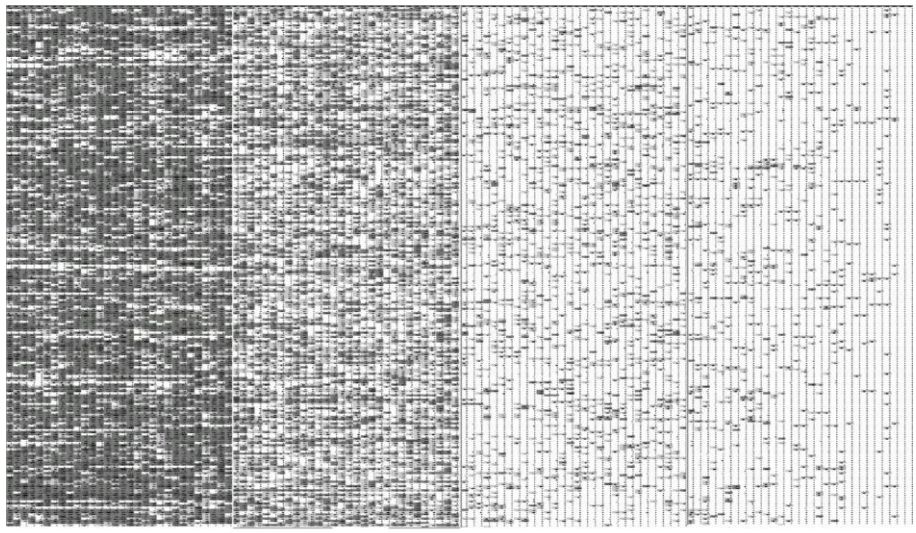


图 10-8　RFM 聚类各类用户热力图

## 4.推荐系统

最后,本例将利用 R 建立推荐系统,对每个人进行 APP 的推荐。推荐系统的基础是人们对 APP 的评分矩阵,矩阵每一行代表一个人,每一列代表一个 APP,该评分矩阵第 $i$ 行第 $j$ 列的元素表示这 30 天内第 $i$ 个人对第 $j$ 个 APP 的有效日均使用时长,本例中将其视为用户对 APP 的“评分”。如果这个人在有效观测天数内没用过该 APP,则矩阵该位置的数据为缺失。由于 APP 数量众多,总量多达几万,绝大多数市场占有率极低,因此我们仅使用 app_class.csv 文件中给出的 4000 多个常用的并且用户数超过 10 个的 APP 进行推荐。最终保留的用户总数为 48139 人,APP 数量为 2542 个。一个人使用的 APP 数量最多只有几十种,因此评分矩阵必然是一个稀疏的矩阵。如果不采用稀疏存储方法,那么需要存储的数字将达到上亿级别,而实际不为 0 的数字只占总元素个数约 2%,因此在 R 当中必须使用稀疏矩阵的方法进行存储。

R 中的 sparseMatrix()函数可以实现稀疏矩阵的存储,同时使用 recommenderlab 包建立推荐系统,recommenderlab 包支持将稀疏矩阵转化为评分矩阵的过程。在建立稀疏矩阵的时候,需要三列数据,即行数、列数、该位置的内容,这些内容可以通过编程从原始数据当中提取出来,获取这个数据集后,通过将用户编号与 APP 编号转化为对应的行名和列名,可以建立稀疏矩阵。

建立了稀疏矩阵后,需先将该矩阵转化为 recommenderlab 包适用的评分矩阵才可以使用。同时,注意到对于不同的 APP,用户总体的使用强度是不同的,而在推荐系统中要求评分矩阵的评分是可以比较的,因此需要将评分按列进行标准化处理,这一步在有些情况下可以省略,因为后面基于物品的协同过滤方法会默认进行这一步处理。

最后使用 recommenderlab 包中的 Recommender()函数建立推荐系统。本例中先使用基于用户的协同过滤方法建立推荐系统,程序为:

```
rec<-Recommender(rm,method="UBCF",
parameter=list(method="pearson",nn=30))
```

其中,method="UBCF"表示使用基于用户的协同过滤方法,参数列表当中的"pearson"表示使用皮尔森相关系数作为相似性度量的依据,nn=30 表示选取前 30 个相似的用户进行评分的预测。

利用 predict()函数来获取预测结果,该函数默认使用 TopN 的方式给出结果,用 n 表示想要选取的推荐个数,利用参数 type="ratings"可以获得估计评分,输出用户前 20 个评分最高的 APP 编号。除基于用户的协同过滤方法外,还可以使用基于物品的协同过滤方法(method="IBCF"),同样获取对第一个用户的前 20 个推荐。

利用两种方法得到的推荐结果见表 10-11(推荐结果已经按 APP 类别排序),可以看到两种方法的推荐结果完全不同,说明虽然都是基于协同过滤的推荐系统方法,但是基于用户还是物品依旧能够造成推荐结果的差异。

表 10-11　第一名用户推荐结果

| UBCF 方法 | | | | | | IBCF 方法 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | APP 编号 | 类别 | No. | APP 编号 | 类别 | No. | APP 编号 | 类别 | No. | APP 编号 | 类别 |
| 1 | 10432 | c | 11 | 22208 | h | 1 | 1851 | b | 11 | 7332 | j |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 9098 | c | 12 | 4010 | i | 2 | 3950 | b | 12 | 1881 | o |
| 3 | 2627 | c | 13 | 18928 | j | 3 | 5450 | d | 13 | 1414 | p |
| 4 | 22643 | c | 14 | 7373 | j | 4 | 22325 | d | 14 | 465 | q |
| 5 | 17445 | c | 15 | 1229 | j | 5 | 7405 | f | 15 | 2670 | q |
| 6 | 22616 | f | 16 | 23547 | k | 6 | 3842 | f | 16 | 2203 | s |
| 7 | 13613 | g | 17 | 7796 | k | 7 | 776 | g | 17 | 9781 | t |
| 8 | 776 | g | 18 | 18000 | p | 8 | 8395 | g | 18 | 23304 | t |
| 9 | 3962 | g | 19 | 3090 | t | 9 | 18535 | h | 19 | 3594 | t |
| 10 | 18504 | h | 20 | 7167 | t | 10 | 4143 | j | 20 | 3155 | t |

IBCF 方法推荐最多的为 t 类别,这一点符合第一位用户对各类 APP 的实际使用情况,他唯独在 t 类别上共花费 334728 秒的时间,在第二个用时最高的 f 类别上只花费了 56766 秒的时间。UBCF 方法推荐最多的为 c 类别,但是实际上该用户不曾在 c 类别上花很多时间,这反映了 UBCF 方法可以根据其他用户的使用习惯发掘出该用户未使用过但是却可能感兴趣的 APP。

在认可推荐系统的强大之前需要对推荐系统的预测效果进行评价,recommenderlab 包中自带了将训练集、测试集分开并进行评估的方法。具体过程如下:evaluationScheme()函数将数据集分为 90%的训练集和 10%的测试集,并给每个观测至少 10 个评分数值,然后利用训练集建立推荐系统,最后对测试集进行预测并比较预测结果和真实结果的差距。由分析结果可以看出,本例数据使用 UBCF 比使用 IBCF 效果略好(见表 10-12)。注意,由于数据分割的随机性,每次运行结果不完全相同,具有一定的波动性。

表 10-12 推荐系统评价指标

| 方法 | RMSE | MSE | MAE |
|---|---|---|---|
| UBCF | 3.45 | 11.89 | 2.77 |
| IBCF | 3.62 | 13.12 | 3.34 |

需要说明的是,R 软件的 recommenderlab 包在人数、物品数较多的情况下建立推荐系统比较慢(如在本例样本规模下基于物品的协同过滤方法在服务器上运算用时 1 个小时左右),在业界实际部署的时候一般不会利用该包直接建立推荐系统,而是在协同过滤和矩阵分解等经典的推荐系统算法的基础上,根据实际数据的特性和需求自行编程。目前无论是学术界还是业界都存在很多对基础的推荐系统(在面对大数据的情况下)的效率、精确度进行改进的方法,如算法组合、运算并行化处理等,其中最著名的活动当数 2009 年举行的 Netflix 百万美元推荐竞赛。

### 10.1.3* 分布式实现

这一部分介绍该案例的分布式实现,包括数据预处理与模型分析两部分。

#### 1.数据预处理与描述分析

由于原始数据是结构化的记录数据,因此可以利用 Hive 进行数据预处理。在此我们仅以实现图 10-1 和图 10-2 的数据准备为例,给出利用 Hive 对数据进行预处理的步骤以及具体语句,读者可在人大出版社提供的网址下载。

#### 2.基于 Spark 的模型分析

数据准备完毕之后,可以利用 Spark 中的 MLlib 对数据进行模型分析。在此我们进行 10.1.2 中单机版的 i 类 APP 的用户行为预测(分类和回归)。预测方法为随机森林。

算法使用的数据是之前利用 Python 处理所得的数据(读者也可自行编写分布式程序对数据进行预处理),运行程序成功后,由结果可知,Spark 分类模型的整体准确率为 82.16%,

相比较单机版随机森林的分类准确率稍高一些;Spark 回归模型的 NMSE 为 0.602,比单机版随机森林回归的 NMSE(0.621)小一些,表示预测效果较好。

接下来,我们使用 Spark 中的 MLlib 进行 K-means 聚类分析,以单机版第一个聚类分析为例。在此不把 $K$ 中心点的选择纳入分布式计算之中,主要还是参考前文,将中心数选为 8,接着基于前面单机版中加 1 后取对数处理的时长数据,利用 Spark 建立 K-means 模型,可以得到算法将观测分为 8 类的结果,每一类的类中心如表 10-13 所示。

表 10-13 分布式聚类各类中心值

| 类别/APP | a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.47 | 2.03 | 5.34 | 3.81 | 2.78 | 7.68 | 6.03 | 7.19 | 2.68 | 1.63 |
| 2 | 0.84 | 0.44 | 1.41 | 4.39 | 0.63 | 4.81 | 3.18 | 0.40 | 0.93 | 0.54 |
| 3 | 2.42 | 1.40 | 5.28 | 4.47 | 6.26 | 7.39 | 5.54 | 0.38 | 2.17 | 1.05 |
| 4 | 1.86 | 1.10 | 3.65 | 4.42 | 6.18 | 6.99 | 4.89 | 0.37 | 2.09 | 0.66 |
| 5 | 2.73 | 1.38 | 5.25 | 4.39 | 0.46 | 7.42 | 5.64 | 0.31 | 2.31 | 0.77 |
| 6 | 5.24 | 3.68 | 5.69 | 4.31 | 5.69 | 8.18 | 6.27 | 0.75 | 4.04 | 1.72 |
| 7 | 4.64 | 2.35 | 5.63 | 4.35 | 0.42 | 7.73 | 6.10 | 0.40 | 3.10 | 1.49 |
| 8 | 0.74 | 0.59 | 4.42 | 4.27 | 0.61 | 6.70 | 4.91 | 0.34 | 0.99 | 0.76 |
| | | | | | | | | | | |
| 类别/APP | k | l | m | n | o | p | q | r | s | t |
| 1 | 3.46 | 0.25 | 0.34 | 1.90 | 1.72 | 3.02 | 2.07 | 0.14 | 0.77 | 4.82 |
| 2 | 0.67 | 0.06 | 0.08 | 0.66 | 0.25 | 0.63 | 0.65 | 0.02 | 0.18 | 0.78 |
| 3 | 2.45 | 0.16 | 0.19 | 1.81 | 0.71 | 1.56 | 2.24 | 0.09 | 0.58 | 6.62 |
| 4 | 1.56 | 0.16 | 0.14 | 1.66 | 0.40 | 1.40 | 1.94 | 0.08 | 0.55 | 0.52 |
| 5 | 2.78 | 0.17 | 0.16 | 1.46 | 0.80 | 2.14 | 2.05 | 0.06 | 0.62 | 0.43 |
| 6 | 3.91 | 0.42 | 0.42 | 2.24 | 2.19 | 4.93 | 3.81 | 0.20 | 1.45 | 3.96 |
| 7 | 3.40 | 0.18 | 0.31 | 1.64 | 1.49 | 2.81 | 2.69 | 0.09 | 0.77 | 6.59 |
| 8 | 1.93 | 0.08 | 0.15 | 1.11 | 0.62 | 1.32 | 1.09 | 0.04 | 0.32 | 6.19 |

对比可以发现单机版的聚类结果与 Spark 聚类结果在各类中心的取值上比较相似,比如,这里的第 2 类对应表 10-6 中的第 1 类等。这里不再具体解释聚类结果。建议读者编写程序输出两种方法的用户聚类的类别标签,然后绘制 8×8 的交叉表,进一步检查两种聚类方法的差异。

最后,使用 Spark 中 MLlib 的 ALS()函数建立推荐系统。ALS 指交替最小二乘法的协同过滤算法,其优点是,相较于 UBCF 与 IBCF,它便于利用分布式架构来实现。使用该算法时,利用前面处理过的推荐系统数据作为输入,运行成功后,对第一位用户推荐前 20 个 APP 的结果如表 10-14 所示。

表 10-14 分布式方法第一位用户推荐结果

| Spark.ALS | | | | | |
|---|---|---|---|---|---|
| No. | APP 编号 | 类别 | No. | APP 编号 | 类别 |
| 1 | 22684 | e | 11 | 4324 | t |
| 2 | 3309 | f | 12 | 6780 | t |
| 3 | 6472 | h | 13 | 7817 | t |
| 4 | 8782 | i | 14 | 9685 | t |
| 5 | 130 | j | 15 | 11112 | t |
| 6 | 7373 | j | 16 | 11540 | t |
| 7 | 21501 | j | 17 | 12822 | t |
| 8 | 23608 | o | 18 | 14198 | t |
| 9 | 2053 | t | 19 | 16470 | t |
| 10 | 2280 | t | 20 | 23168 | t |

由结果可知,对 t 类 APP 的推荐力度最大,为 12 次,这与该用户钟爱 t 类 APP 有很大关系;接着是 j 类,为 3 次。最后利用单机版建立模型时生成的 90%的训练集与 10%的测试集样本来计算模型的误差,对 RMSE 进行对比可以发现,2.49 比前面 UBCF 与 IBCF 方法的推荐系统都要小,但总体是相近的。