**西安电子科技大学2019年数学建模校内赛**

**承 诺 与 产 权 转 让 书**

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们同意将参赛论文以及支撑材料中的所建模型、算法以及程序产权归属西安电子科技大学以及合作单位共有。特别的，B题参赛论文以及支撑材料中的相应产权西安电子科技大学拥有50%，合作单位享有50%。2019年数学建模校内赛竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号是（从A/B中选择一项填写）： B

**西安电子科技大学2019年大学生数学建模校内赛**

**评 阅 专 用 页**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 评阅人1 | 评阅人2 | 评阅人3 | 总评 |
| 成绩 |  |  |  |  |

使用神经网络与AutoML基于APP时序埋点数据的用户预测

摘要

随着信息时代的发展，互联网+产业越来越普及。在信息时代中，研究如何利用网络大数据预测用户行为对企业，政府等都有着积极意义。本文基于BP神经网络、朴素贝叶斯、随机森林等机器学习算法，从App的大量用户时序埋点数据入手，考虑用户行为相互独立的情况，分析影响用户消费行为的主要因素并对用户是否购买进行了合理预测，给出了比较合理的方案。

针对问题一影响用户消费行为的因素涉及面广，逻辑性不强，很难集中地用数学方式进行表述与分析的情况，本文采用了选取数据样本的主要特性来构建BP神经网络分类预测模型，选取了一部分训练集来对模型进行训练，最终对测试集进行检验，经评估模型性能良好。

针对问题二，本文使用了Python TPOT 搭建的AutoML模型从而得到最优算法以及最优参数，有广泛的普适性，实现了数据自学习。AutoML模型选用了朴素贝叶斯(GaussianNB)和随机森林(RandomForest)两种最优算法，本文对这两种机器学习算法进行了比较并对数据进行了分析预测，最终选择了随机森林算法作为预测模型，经评估模型性能良好。

本文构建的模型，可以较好地实现对用户行为的预测，并有较高的测验成功率，可以帮助我们对App用户的行为进行大致预测，并最终实现资源的高效分配。

关键词：BP神经网络、二分类、AutoML、TPOT、朴素贝叶斯、随机森林

一、问题重述

1.1问题的背景

随着信息时代的大潮涌动，互联网+产业越来越普及，人们更多地依赖于网络APP从事相关活动。许多消费行为都可以在APP软件上直接操作，例如购物消费、点评评价、信贷消费等。用户在(手机端、电脑端等)APP上的活动，如注册、浏览、查询、是否按期还款、是否及时评价等诸多类型，体现了用户的不同行为习惯、行为偏好，同时不同用户从事相关活动的时间、顺序各不相同，同一用户在不同时间段从事同一活动的行为也有差异。这些不同网络行为、行为时间等行为表现与用户最终的某个决定——是否实际消费、是否会按期偿还贷款、是否满意某商业行为等密切相关。

1.2需要解决的问题

研究人员通过对大数据的研究发现，这些时序埋点等数据反映了用户的兴趣与需求，如果能够将其深入挖掘并且合理利用，可以用来指导商业APP的运营。若能提前预测用户下一步的行为，甚至提前得知用户卸载、流失的可能性，则能更好地指导产品的优化以及用户的精细化运营。需要解决的问题有：

1.基于用户APP行为数据，建立预测用户行为的数学模型，并针对提供的数据，评估模型的性能。

2。设计适用于这种时序埋点数据的、无人工干预的AutoML模型，需要考虑数据应用场景些许变化、埋点类别变化等因素。编写可读取其他数据作为训练数据训练、及其相关测试数据测试的可执行程序。

二、问题的分析

2.1问题一的分析

问题一要求我们根据某APP应用环境采集的大量用户的时序埋点数据及其行为标签，基于用户APP的行为数据，建立预测用户行为的数学模型并对其进行评估。由于用户数据量庞大且复杂，无法直观地看出某个数据对用户行为变量的贡献率是大是小（即贡献率的大小没有一个尺度来衡量），我们采用BP神经网络进行建模。

2.2问题二的分析

问题二要求建立一个AutoML算法，在给定数据和度量标准之后，通过算法自动选择出效果最好的模型。为了快速高效地建立模型，我们使用了Python TPOT库建立AutoML，传入数据后对结果进行分析，选择最优模型。

三、模型基本假设

（1）App用户时序埋点数据能较真实反映用户个人操作的真实情况。

（2）用户时序数据与其行为标签具有紧密联系，可以通过模型找到两者的关系。

（3）对之前搜集的时序埋点数据能较好适用于对其它用户行为的分析，建立的模型适用面广泛。

（4）埋点捕获时间、埋点捕获情况对用户是否进行再次购买影响较大。

四、符号说明

|  |  |
| --- | --- |
| 符号 | 说明 |
| W | 链接权重 |
| b | 偏置 |
|  | 学习率 |
| N0 | 无再次购买用户埋点的期望 |
| N1 | 再次购买用户埋点的期望 |
| h | 输出 |
| Wj,k | 权重改变值 |
| P | 概率 |

五、模型的建立与求解

5.1构建模型前的准备和数据预处理：

对于时序埋点数据，我们分别从时间次序与埋点两方面进行数据的分析：

（1）埋点数据分析

从已知train\_val\_data集中随机取样（1000个用户），并统计用户埋点行为可得下图（图1）：

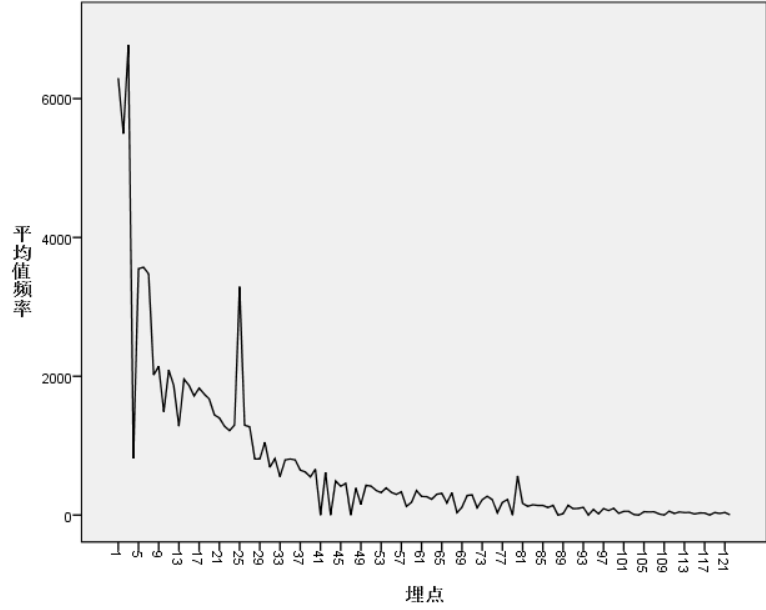


图1 埋点数据分析图

由图可以分析用户行为的分布，在前35个埋点中具有更高的频率，我们可以认为这些埋点对用户行为（如消费活动）具有重要影响。

接下来我们分析了有再次购买用户的埋点数据与无再次购买用户数据，计算用户各埋点行为次数的样本期望来反映频率，得下图（图2）：

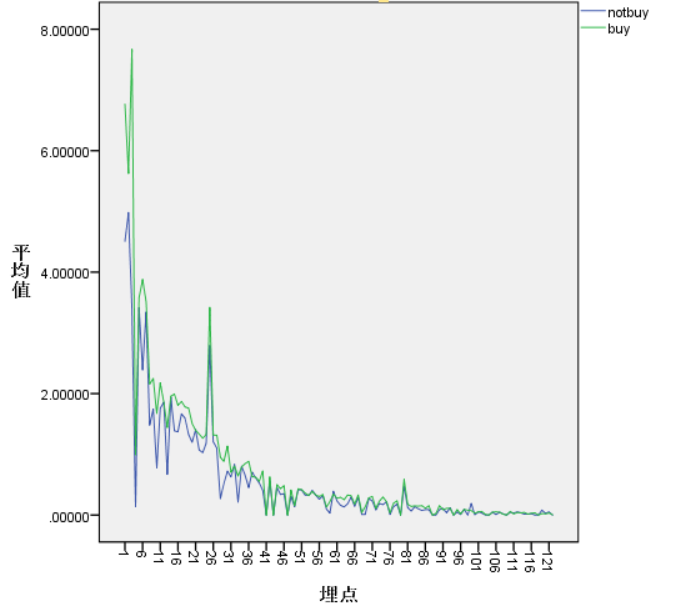


图2

设N0为无再次购买用户埋点的期望，N1为再次购买用户埋点的期望，可得下表（表1）：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | ... |
| N0 | 4.509524 | 4.980952 | 3.361905 | 0.142857 | … |
| N1 | 6.76359 | 5.624526 | 7.667509 | 0.992415 | … |

表1

可以明显看到，有再次购买用户与无再次购买用户的一些埋点行为有很大差异。我们设г为N0 / N1 的比值(0<г<1),通过调节г对数据进行筛选，可以得出影响消费行为程度较大的埋点。

（2）时序数据分析：

编写程序将entry\_time 的UNIX\_时间戳 通过算法 转换为 67个时间点，并将用户的消费行为频率用图表形式表示：

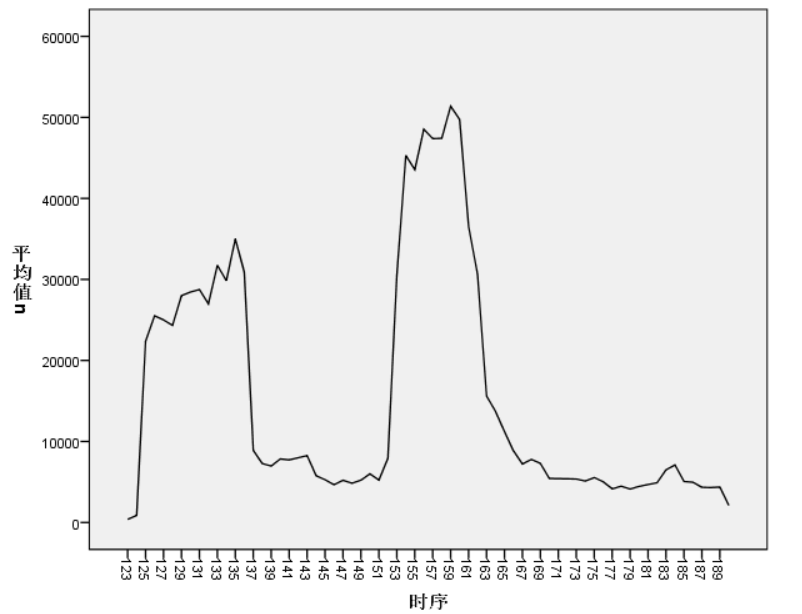


图 3

由图中我们可以观察到峰值，可以分析出时间对用户消费行为具有较强影响。于是，我们在构建模型时考虑埋点与时序对用户是否再次购买的影响。

5.2问题一：基于用户APP行为数据 ，建立预测用户行为的神经网络模型

5.2.2模型的建立

App记录了样本创建时间、样本实际完成时间、样本期望完成时间、脱敏IP地址、脱敏设备唯一标识符、APP log埋点被抓获时间、APP log脱敏埋点名称。根据数据分析，我们认为样本时长、埋点被抓获情况对用户是否进行再次购买影响较大，因此从数据中选取样本创建时间、样本实际完成时间、APP log脱敏埋点名称作为输入变量，用户Label（1代表再次购买，0代表无再次购买）作为输出变量构建神经网络模型如下图：

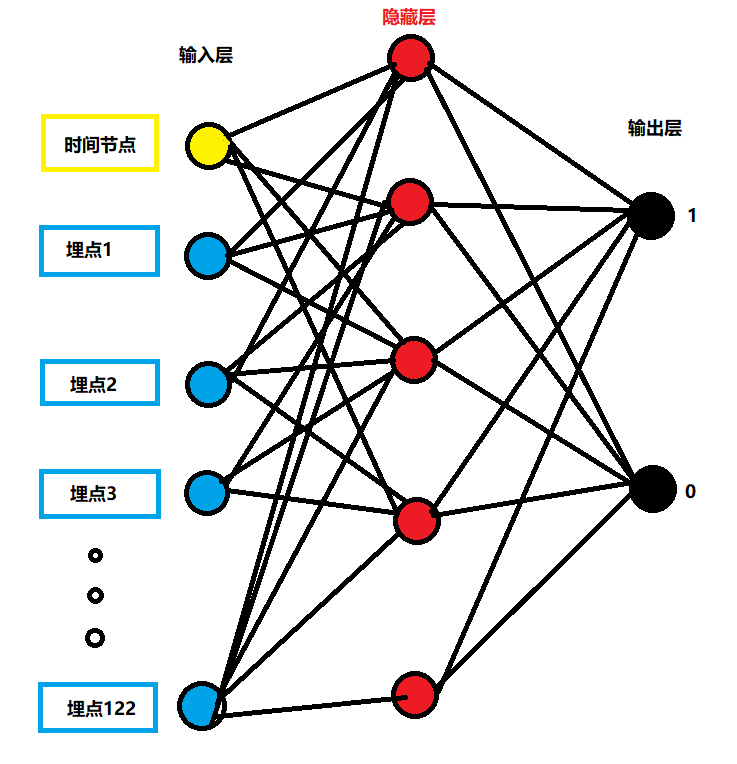


图4 BP神经网络模型图

**输入层**：

输入层共190个节点，前122个节点的输入变量表示用户激活各埋点的次数，后67给节点的输入变量表示用户在该时间点消费行为的的次数。

为保证输入变量节点不会因数据过大而支配整个神经网络，我们对其进行归一化处理：X\*= 并赋予特殊的权重。

将输入引入神经网络。

**隐藏层**：

其中是激发函数，可以取不同形式。本文采用Sigmoid函数，通过随机生成的正态分布的权重矩阵连接输入层与输出层。



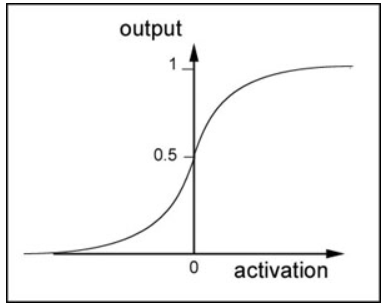


图5 Sigmoid函数图像

**输出层**：

输出层共两个节点，分别代表label为0和label为1，两个节点分别得到一个输出，通过比较两个节点输出大小来预测样本的label，实现二分类。

**反向传播：**

设W为权重，b为偏置，为学习率。

代价函数：J(W,b;x,y)=||hW,b(x)-y2||2

给定一个样例(x,y)，我们首先进行“前向传导”运算，计算出网络中所有的激活值，包括 \textstyle h_{W,b}(x) 的输出值。之后，针对第L层的每一个节点i，我们与训练集的标签对比计算出误差，根据误差调整权重与偏置。



分析每一层神经网络，可得：





其中基于以下公式，偏差逐步反传：





得出各因素的处理后权值：

 ****

**训练集与数据集**：

从样本中选取1000个id作为测试集，其余作为训练集进行训练。

5.2.2模型求解：

神经网络是一种适宜处理具有残缺结构和能够分析含有错误成分的算法,它能够在信息含糊、不确定、不完整、存在矛盾及假象等复杂环境中处理分析数据；并且神经网络所具有的自学能力，使得传统数学算法应用最为困难的有效数据获取工作，转换为网络的变结构调节过程,从而大大方便了各种不同应用对象的建模与分析，进而可以对一些复杂问题做出合理的判断决策,做出有效的预测和估计。

我们将时序埋点数据对每个id激活埋点的情况进行统计，使每个id对应一个行向量，每个埋点对应1个列向量，构成输入矩阵如下表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | F1 | F2 | … |
| 52263074 | 3 | 0 | … |
| 52263431 | 8 | 6 | … |
| 52263529 | … | … | … |
| … | … | … | … |

表2 输入矩阵

将Model\_Test\_Data传入模型得到预测数据。（见附件“第一题.txt”）

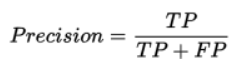
**5.2.3结果检验分析：**

模型评估：

（1）查准率(Precision)

关注预测为正样本的数据(可能包含负样本)中,真实正样本的比例。

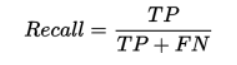
计算公式：



（2）查全率（Recall）

关注真实正样本的数据(不包含任何负样本)中,正确预测的比例。

计算公式：



### （3）分类模型的评估方法-F分数(F-score)

### 查准率(Precision)和查全率(Recall)评估指标,理想情况下做到两个指标都高当然最好，但一般情况下，Precision高，Recall就低，Recall高，Precision就低。所以在实际中常常需要根据具体情况做出取舍，例如一般的搜索情况，在保证召回率的条件下，尽量提升精确率。而像癌症检测、地震检测、金融欺诈等，则在保证精确率的条件下，尽量提升召回率。引出了一个新的指标F-score,综合考虑Precision和Recall的调和值。

### 当β=1时，称为 F1-score

### F.png

### 

根据我们的模型所得出的结果，经计算得到：

*P = 0.6643*

*R = 0.9545*

*F1-score = 0.7846*

**5.3问题二：使用Python tpot库构建AutoML模型**

**5.3.1模型的建立**

Python TPOT（The Tree-Based Pipeline Optimization Tool）库的目标是通过将管道的灵活表达树 (Flexible Expression Tree) 表示与诸如遗传编程 (Genetic Programming) 的随机搜索算法相结合来自动化机器学习管道的构建。 TPOT 使用基于 Python 的 scikit-learn 库作为其机器学习基础库，使程序编写者无需尝试每一种机器学习方法而找到最优方法。

构建AutoML的核心代码如下：

pipeline\_optimizer = TPOTClassifier(generations=5, population\_size=20, cv=5，random\_state=42, verbosity=2)

pipeline\_optimizer.fit(X\_train, y\_train)

通过AutoML模型得到两种较好的算法：朴素贝叶斯与随机森林。

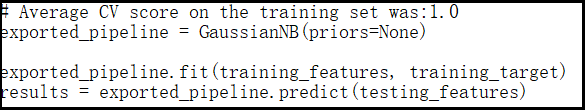


图6 通过AutoML模型得到的朴素贝叶斯结果

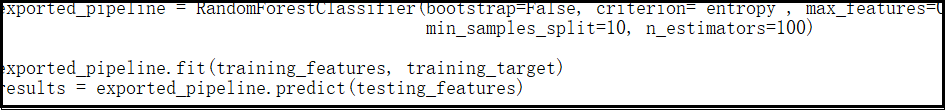


图7 通过AutoML模型得到的随机森林结果

**5.3.2 朴素贝叶斯(Naïve Bayes)**：

1.朴素贝叶斯分类模型

（1）条件概率和乘法定理

在事件*A*已经发生的条件下，时间*B*发生的概率，称为事件*B*在给定事件*A*的条件概率(也称为后验概率)，记作*P(B|A)*。相应的，*P(A)*成为无条件概率（也称为先验概率）。条件概率可以由下式进行计算：

*P(B|A) =*

由条件概率可求得概率的乘法定理：

*P(AB) = P(B|A)P(A)*

对于n个事件*A1,A2,…,An，n≥2*, 则有：

*P(A1,A2,…,An) = P(An | A1,A2,…,An-1) P(An-1 | A1,A2,…,An-2)P(A2 | A1)P(A1)*

（2）全概率公式和贝叶斯定理

设试验*E* 的样本空间为S,A为E的事件，*B1,B2,…,Bn* 为S的一个划分，且P(*Bi*)>0(i=1,2,...n),则：

*P(A)=P(A|B1)P(B1) + P(A|B2)P(B2)+…+ P(A|Bn)P(Bn)*

*=*

上式成为全概率公式。

设试验*E*的样本空间为S,A为E的事件，B1,B2,…,Bn 为S的一个划分，且*P(A)>0 , P(B)>0(i=1,2,…,n)* ,则由条件概率的定义和全概率公式：

*P(Bi|A) =*

上式称作贝叶斯定理。

（3）朴素贝叶斯分类模型

朴素贝叶斯(Naïve Bayes)分类器是一种简单而高效的分类器，可以与决策树和经过挑选的神经网络分类器相媲美。朴素贝叶斯分类器(Naïve Bayes Classifier,NBC)则是贝叶斯模型的一种。

设训练元组为  *x={* ***A****1 ,* ***A****2,* ***A****3****, … A****n}* ,其中**α**i 为x 的一个属性，在该模型中，所有属性都独立于类别变量C，每个属性都有唯一的父节点，这样将大大简化贝叶斯网络构建的过程。

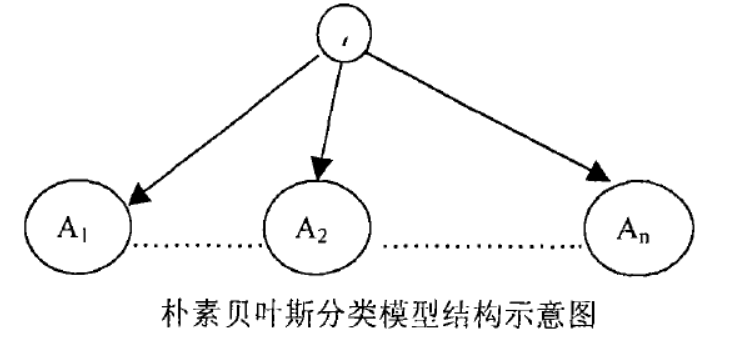


图8 朴素叶斯分类模型结构示意图

2.朴素贝叶斯分类的工作过程

(1)每个数据样本用一个n维特征向量*X*={ *x1,x2,…,xn*}表示，分别描述对n个

属性*A1,A2,…,An* 样本的n个度量。

(2)假定有m个类*C1,C2,…,Cm* 。给定一个未知的数据样本X(即没有类标号)，分类法将预测X属于具有最高后验概率(条件X下)的类。朴素贝叶斯分类将未知的样本分配给类C*i* ,当且仅当：

*P(Ci|X) > P(Cj|X) , 1≤i,j≤m，j≤m , j≠m , j≠ i*

这样，最大化P(*C i*|X), P(*Cj* |X)最大的类*C I* 称为最大后验假定，根据贝叶斯定理：

*P(C i |X)=*

(3) 给定具有许多属性的数据集，可以做类条件独立的朴素假定。对于给定样本的类标号，假定属性值相互条件独立，则：

*P(X|Ci) =*

概率*P(x1|Ci) , P(x2|Ci),…, P(xn|Ci)*可以由训练样本估计。

**5.3.3 随机森林(RandomForest)**：

随机森林算法是由多棵分类回归树（classification and regression tree，CART）组合构成的新型机器学习算法, 是一种Bagging类型的集成算法，通过组合多个弱分类器，最终结果通过投票或取均值，使得整体模型的结果具有较高的精确度和泛化性能。

随机森林学习速度很快，在处理很大数据时依然非常高效，因此适用于对App时序数据这样的大数据进行快速分析。在第一问中，我们已进行了数据处理，形成了用于训练随机森林模型的训练数据库，为每个训练集建立分类回归树，产生由Ntree棵CART决策树组成的森林，在每棵树生长过程中，从全部M个特征变量 中随机抽选m个（m≤M），在这m个属性中根据 Gini系数最小原则选出最优属性进行内部节点分支；最后，集合Ntree棵决策树的预测结果，对于分类问题，采用结果的众多树投票结果的众数作为分类的结果，即label值。

具体实现过程如下：

（1）原始训练集为N，应用bootstrap法有放回地随机抽取k个新的自助样本集，并由此构建k棵分类树，每次未被抽到的样本组成了k个袋外数据；

（2）设有mall个变量，则在每一棵树的每个节点处随机抽取mtry个变量(mtry n mall)，然后在mtry中选择一个最具有分类能力的变量，变量分类的阈值通过检查每一个分类点确定；

（3）每棵树最大限度地生长, 不做任何修剪；

（4）将生成的多棵分类树组成随机森林，用随机森林分类器对新的数据进行判别与分类，分类结果按树分类器的投票多少而定。

**5.3.4模型求解：**

将Model\_Test\_Data传入模型得到预测数据。（见“第二题.txt”）

**5.3.5结果检验分析**

我们根据AutoML得到了AI所认为的最好的算法，我们以此使用朴素贝叶斯中的GaussianNB算法预测数据。得到的预测图如下：

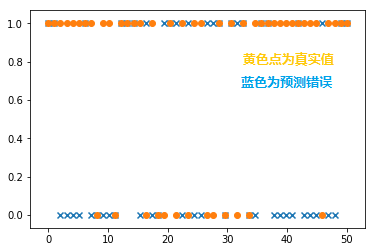


图9 朴素贝叶斯

使用随机森林模型对测试机进行预测得到的结果如下图：

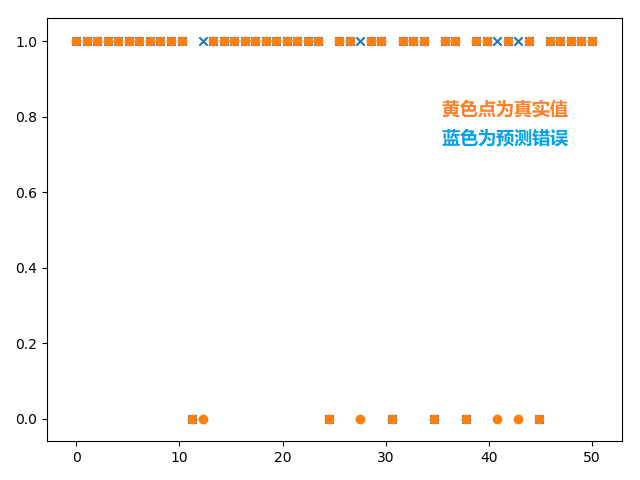


图10 随机森林

决策树的预测结果明显好于朴素贝叶斯，抽测的五十个样本中仅有四条预测错误。

六、优缺点分析

优点：

**BP神经网络：**

（1）非线性映射能力：问题一数学模型的建立运用了BP神经网络，具有较强的非线性映射能力，适合解决如本文涉及的“基于时序埋点数据的用户行为预测” 等内部机制复杂的问题。

（2）自学习和自适应能力：神经网络还具有高度的的自学习与自适应能力，将学习内容记忆于网络的权值之中。

（3）容错能力：BP神经网络在其局部或部分神经元受到破坏后对全局的训练结果不会造成很大的影响，具有一定的容错能力。

**朴素贝叶斯：**

(1) 朴素贝叶斯模型有稳定的分类效率。

(2) 对小规模的数据表现很好，能处理多分类任务，适合增量式训练，尤其是数据量超出内存时，可以一批批的去增量训练。

(3) 对缺失数据不太敏感，算法也比较简单。

**随机森林：**

（1）在当前的很多数据集上，相对其他算法有着很大的优势，表现良好。

（2）它能够处理很高维度（feature很多）的数据，并且不用做特征选择（注：特征子集是随机选择的）

（3）实现比较简单，训练速度快，容易做成并行化方法

（4）对于不平衡的数据集来说，它可以平衡误差，如果有很大一部分的特征遗失，仍可以维持准确度。

缺点：

**BP神经网络：**

（1）BP神经网络的缺点是它的算法会倾向于观测值较多的类别：由于训练集的样本中label为1的占大多数，因此训练好的神经网络会偏向于预测label为1，会影响预测结果的准确性。

（2）BP神经网络预测能力和训练能力的矛盾问题：一般情况下，训练能力差时，预测能力也差，随着训练能力的提高，预测能力会得到提高，但存在一个极限。当达到此极限时，随训练能力的提高，预测能力反而会下降，即出现所谓“过拟合”现象。

**朴素贝叶斯：**

(1) 需要知道先验概率，且先验概率很多时候取决于假设，假设的模型可以有很多种，因此在某些时候会由于假设的先验模型的原因导致预测效果不佳。

(2) 由于我们是通过先验和数据来决定后验的概率从而决定分类，所以分类决策存在一定的错误率。

**随机森林：**

（1）随机森林在某些噪音较大的分类或回归问题上会过拟合

（2）对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响

七、模型拓展

本文提供的模型基本能够解决企业预测用户行为的需求。企业可以通过此模型大致预测用户行为，对资源进行有目的的分配，从而节约资金与资源，提高销量。通过对本模型的求解反映了对用户影响较大的因素：即时间次序、埋点数据。为此企业应该根据这些因素之间的联系作出相应的调整以推动用户的消费行为。

除此之外，还可以将本模型推广到其他大数据处理分析预测的问题中，例如在交通旅游领域中，可以通过基站定位技术，得到类似本题的时序埋点数据，政府可以对每个车站、机场、道路交通各时段的流量规律信息，进行针对性的安防部署，景点可以根据游客的浏览路径与时间，决定开放时间及区域。

用户的行为举动往往带有一定的目的性与一致性，通过本文模型可以一定程度上帮助我们了解用户行为趋势。

八、参考文献资料

**参考文献：**

《机器学习》周志华2016版 2019-5-1

《Python神经网络编程》 Tariq Rashid  2019-5-1

《随机森林模型在分类与回归分析中的应用》李欣海 2019-5-1

**参考网址：**

<http://xueshu.baidu.com/usercenter/paper/show?paperid=9e39427d8c4c870c28d1fc59d8cd9e2a&site=xueshu_se> 2019-5-1

<http://xueshu.baidu.com/usercenter/paper/show?paperid=8eae48e09ead0274635120acb316b740&site=xueshu_se> 2019-5-1

<https://www.cnblogs.com/nullzx/p/9234058.html> 2019-5-1

<https://blog.csdn.net/mr_muli/article/details/84481458> 2019-5-2

<https://blog.csdn.net/tz_zs/article/details/73477870> 2019-5-2

<https://blog.csdn.net/j904538808/article/details/80192091> 2019-5-3

九、附录

**第一题数据处理代码#data.py**

import numpy as np

import csv

# 读id

idx = np.loadtxt(r"C:\Users\14531\Desktop\数据处理.CSV", delimiter=",", usecols=0)

idx = idx.astype(np.int32)

# 截掉左侧的f

def mystr(s):

s = s.lstrip("f")

return s

csv\_file = open(r"C:\Users\14531\Desktop\point.CSV") # 读埋点

csv\_reader\_lines = csv.reader(csv\_file)

date\_PyList = []

for one\_line in csv\_reader\_lines:

date\_PyList.append(one\_line[0])

date\_ndarray0 = list(map(mystr, date\_PyList))

fx = list(map(int, date\_ndarray0))

csv\_file.close()

# 创建二维数组，id为行数，埋点为列数

end = np.zeros((53441203, 123), dtype=np.int32)

for index, f in enumerate(fx):

temp = idx[index]

end[temp][f] += 1

end[temp][0] = 1 # 标记已记的点

label = np.zeros((60000000, 1), dtype=np.int64) # 存label的二维数组

labelx = np.loadtxt(r"C:\Users\14531\Desktop\Label.csv", delimiter=",", usecols=(0, 1))

for i in range(len(labelx)):

temp = int(labelx[i][0])

label[temp][0] = labelx[i][1]

# 读时间

timex = np.loadtxt(r"C:\Users\14531\Desktop\time归一.CSV", delimiter=",", usecols=(0, 1))

timex = timex.astype(np.float64)

r'''

# 测试

# 读id

testidx = np.loadtxt(r"C:\Users\14531\Desktop\测试数据处理.CSV", delimiter=",", usecols=0)

testidx = testidx.astype(np.int32)

# 截掉左侧的f

def mystr(s):

s = s.lstrip("f")

return s

csv\_file = open(r"C:\Users\14531\Desktop\测试point.CSV") # 读埋点

csv\_reader\_lines = csv.reader(csv\_file)

date\_PyList = []

for one\_line in csv\_reader\_lines:

date\_PyList.append(one\_line[0])

date\_ndarray0 = list(map(mystr, date\_PyList))

testfx = list(map(int, date\_ndarray0))

csv\_file.close()

# 创建二维数组，id为行数，埋点为列数

testend = np.zeros((55727896, 123), dtype=np.int8)

for index, f in enumerate(testfx):

temp = testidx[index]

testend[temp][f] += 1

testend[temp][0] = 1 # 标记已记的点

testtimex = np.loadtxt(r"C:\Users\14531\Desktop\测试time归一.CSV", delimiter=",", usecols=(0, 1))

testtimex = testtimex.astype(np.float64)

'''

**第一题神经网络代码：**

import numpy # 矩阵

import scipy.spatial # 激活函数1/1+(e^-x)

from matplotlib import pyplot

import scipy.misc

from data import end, label, timex, testend, testtimex

import time

import csv

import sys

class neutralnetwork: # 神经网络类

def \_\_init\_\_(self, inputnodes, hiddennodes, outputnodes, learngrates):

self.inputnode = inputnodes # 节点数

self.hiddennode = hiddennodes

self.outputnode = outputnodes

# self.weighih = numpy.random.normal(0.0, pow(self.hiddennode, 0.5), (self.hiddennode, self.inputnode))

self.weighih = numpy.ones([self.hiddennode, self.inputnode], int)

# 生成权重input—hidden的正态分布矩阵

self.weighho = numpy.random.normal(0.0, pow(self.outputnode, 0.5), (self.outputnode, self.hiddennode))

# 生成权重hidden—output的正态分布矩阵

self.learnrate = learngrates

self.active\_fun = lambda x: numpy.maximum(x , 0) # 激活函数

def connect(self, inputs): # 连接神经网络

inputinput = numpy.array(inputs, ndmin=2).T # 矩阵转置

hiddeninput = numpy.dot(self.weighih, inputinput) # 隐藏层的输入矩阵 dot为点乘

hiddenoutput = self.active\_fun(hiddeninput) # 隐藏层的输出矩阵

outputinput = numpy.dot(self.weighho, hiddenoutput) # 输出层的输入矩阵

outputoutput = self.active\_fun(outputinput) # 输出层的输出矩阵

return outputoutput

def train(self, inputs, targets): # 训练神经网络

input = numpy.array(inputs, ndmin=2).T

target = numpy.array(targets, ndmin=2).T

hiddeninput = numpy.dot(self.weighih, input) # 隐藏层的输入矩阵

hiddenoutput = self.active\_fun(hiddeninput) # 隐藏层的输出矩阵

outputinput = numpy.dot(self.weighho, hiddenoutput) # 输出层的输入矩阵

outputoutput = self.active\_fun(outputinput) # 输出层的输出矩阵

outputerror = target - outputoutput # 输出的误差

hiddenerror = numpy.dot(self.weighho.T, outputerror)

# Wj,k的改变量=学习率\*k的误差\*sigmoid（Ok）\*（1-sigmoid（Ok))点乘Oj的转置

self.weighho += self.learnrate \* numpy.dot((outputerror \* outputoutput \* (1 - outputoutput)),

numpy.transpose(hiddenoutput))

self.weighih += self.learnrate \* numpy.dot((hiddenerror \* hiddenoutput \* (1 - hiddenoutput)),

numpy.transpose(input))

# 调节权重 其中transpose为转置

time.clock()

# 训练

input\_nodes = 123 # 122个输入节点

hidden\_nodes = 300

output\_nodes = 2 # 两个输出节点

learn\_rate = 0.05

n = neutralnetwork(input\_nodes, hidden\_nodes, output\_nodes, learn\_rate)

j0 = 0

for index in range(40000000, len(end)):

yangben = end[index]

if yangben[0] == 1 & j0 > 1000: # end中有意义的样本

onetime = 0

j0 += 1

for one in timex:

if abs(one[0] - index) < 0.1:

onetime = int(one[1] \* 10)

break

ma=max(yangben)

yangben=list(map(lambda x:x/ma,yangben))

inputarr = numpy.asfarray(yangben[1:])

inputarr = numpy.append(inputarr, onetime)

targets = numpy.zeros(2) + 0.01

temp = label[index][0]

targets[temp] = 0.99

n.train(inputarr, targets)

elif yangben[0] == 1:

j0 += 1

print("训练完成", "时间", time.clock())

# 测试成功率

scorescard = []

j = 0

for index1 in range(48876790, len(end)):

yangben = end[index1]

if yangben[0] == 1:

onetime = 0

for one in timex:

if abs(one[0] - index1) < 0.1:

onetime = int(one[1] \* 10)

break

inputarr = numpy.asfarray(yangben[1:])

inputarr = numpy.append(inputarr, onetime)

correctnum = label[index1][0]

outputarr = n.connect(inputarr)

outlabel = numpy.argmax(outputarr)

if outlabel == correctnum:

scorescard.append(1)

else:

scorescard.append(0)

j += 1

if j > 1000:

break

scorescardarr = numpy.asarray(scorescard)

r=scorescardarr.sum() / scorescardarr.size

print("成功率=", r, "时间", time.clock())

if r<0.7:

print("成功率太低")

sys.exit()

# 得到测试数据标签

answer = numpy.zeros([15000, 2],int)

i = 0

#

for index1 in range(53452080,55727896):

yangben = testend[index1]

if yangben[0] == 1:

onetime = 0

for one in testtimex:

if abs(one[0] - index1) < 0.1:

onetime = int(one[1] \* 10)

break

inputarr = numpy.asfarray(yangben[1:])

inputarr = numpy.append(inputarr, onetime)

outputarr = n.connect(inputarr)

outlabel = numpy.argmax(outputarr)

answer[i][0] = index1

answer[i][1] = outlabel

i += 1

if i%100==0:

print(i)

with open(r"C:\Users\14531\Desktop\1.csv", "w",newline="")as f:

writer = csv.writer(f)

for row in answer:

writer.writerow(row)

**第二题AutoML代码：**

import numpy as np

import pandas as pd

from sklearn import model\_selection

from sklearn.ensemble import RandomForestClassifier

from tpot import TPOTClassifier

import csv

# 读id

idx = np.loadtxt(r"C:\Users\14531\Desktop\数据处理.CSV", delimiter=",", usecols=0)

idx = idx.astype(np.int32)

# 截掉埋点左侧的f

def mystr(s):

s = s.lstrip("f")

return s

csv\_file = open(r"C:\Users\14531\Desktop\point.CSV") # 读埋点

csv\_reader\_lines = csv.reader(csv\_file)

date\_PyList = []

for one\_line in csv\_reader\_lines:

date\_PyList.append(one\_line[0])

date\_ndarray0 = list(map(mystr, date\_PyList))

fx = list(map(int, date\_ndarray0))

csv\_file.close()

label = np.zeros((60000000, 1), dtype=np.int8) # 存label的二维数组

labelx = np.loadtxt(r"C:\Users\14531\Desktop\Label.csv", delimiter=",", usecols=(0, 1))

for i in range(len(labelx)):

temp = int(labelx[i][0])

label[temp][0] = labelx[i][1]

# 创建二维数组，id为行数，埋点为列数

end = np.zeros((10000, 123), dtype=np.int8)

k = 0

m = 52263074

for index, f in enumerate(fx):

temp = idx[index] # temp为id

end[k][f] += 1

end[k][0] = label[temp][0]

if m != temp:

k += 1

m = temp

if k >= 10000:

break

print(end[:][1])

X\_train, X\_test, y\_train, y\_test = model\_selection.train\_test\_split(end[:, 1:], end[:, 0],

train\_size=0.75, test\_size=0.25)

pipeline\_optimizer = TPOTClassifier(generations=5, population\_size=20, cv=5,

random\_state=42, verbosity=2)

pipeline\_optimizer.fit(X\_train, y\_train)

print(pipeline\_optimizer.score(X\_test, y\_test))

pipeline\_optimizer.export("code.py")

**第二题朴素贝叶斯代码：**

import numpy as np

import pandas as pd

from sklearn import datasets, naive\_bayes

from sklearn.model\_selection import train\_test\_split

import csv

# 读id

idx = np.loadtxt(r"C:\Users\14531\Desktop\数据处理.CSV", delimiter=",", usecols=0)

idx = idx.astype(np.int32)

# 截掉埋点左侧的f

def mystr(s):

s = s.lstrip("f")

return s

csv\_file = open(r"C:\Users\14531\Desktop\point.CSV") # 读埋点

csv\_reader\_lines = csv.reader(csv\_file)

date\_PyList = []

for one\_line in csv\_reader\_lines:

date\_PyList.append(one\_line[0])

date\_ndarray0 = list(map(mystr, date\_PyList))

fx = list(map(int, date\_ndarray0))

csv\_file.close()

label = np.zeros((60000000, 1), dtype=np.int8) # 存label的二维数组

labelx = np.loadtxt(r"C:\Users\14531\Desktop\Label.csv", delimiter=",", usecols=(0, 1))

for i in range(len(labelx)):

temp = int(labelx[i][0])

label[temp][0] = labelx[i][1]

# 创建二维数组，id为行数，埋点为列数

end = np.zeros((10000, 123), dtype=np.int8)

k = 0

m = 52263074

for index, f in enumerate(fx):

temp = idx[index] # temp为id

end[k][f] += 1

end[k][0] = label[temp][0]

if m != temp:

k += 1

m = temp

if k >= 10000:

break

x\_train, x\_test, y\_train, y\_test = train\_test\_split(end[:, 1:], end[:, 0],

train\_size=0.75, test\_size=0.25)

cls = naive\_bayes.GaussianNB()

cls.fit(x\_train, y\_train)

print('高斯贝叶斯分类器')

print('Training Score: %.2f' % cls.score(x\_train, y\_train))

print('Testing Score: %.2f' % cls.score(x\_test, y\_test))

**第二题随机森林代码：**

import numpy as np

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

import csv

# 读id

idx = np.loadtxt(r"C:\Users\14531\Desktop\数据处理.CSV", delimiter=",", usecols=0)

idx = idx.astype(np.int32)

# 截掉埋点左侧的f

def mystr(s):

s = s.lstrip("f")

return s

csv\_file = open(r"C:\Users\14531\Desktop\point.CSV") # 读埋点

csv\_reader\_lines = csv.reader(csv\_file)

date\_PyList = []

for one\_line in csv\_reader\_lines:

date\_PyList.append(one\_line[0])

date\_ndarray0 = list(map(mystr, date\_PyList))

fx = list(map(int, date\_ndarray0))

csv\_file.close()

label = np.zeros((60000000, 1), dtype=np.int8) # 存label的二维数组

labelx = np.loadtxt(r"C:\Users\14531\Desktop\Label.csv", delimiter=",", usecols=(0, 1))

for i in range(len(labelx)):

temp = int(labelx[i][0])

label[temp][0] = labelx[i][1]

end = np.zeros((11000, 123), dtype=np.int8)

k = 0

m = 52263074

index=[5]

for index, f in enumerate(fx):

temp = idx[index] # temp为i

end[k][f] += 1

end[k][0] = label[temp][0]

if m != temp:

k += 1

m = temp

if k >= 10000:

break

X\_train, X\_test, y\_train, y\_test = train\_test\_split(end[:, 1:], end[:, 0],

train\_size=0.95, test\_size=0.05)

exported\_pipeline = RandomForestClassifier(bootstrap=False, criterion="entropy", max\_features=0.8, min\_samples\_leaf=10,

min\_samples\_split=10, n\_estimators=100)

exported\_pipeline.fit(X\_train, y\_train)

print(exported\_pipeline.score(X\_test, y\_test))

# 读id

idx = np.loadtxt(r"C:\Users\14531\Desktop\test数据处理.csv", delimiter=",", usecols=0)

idx = idx.astype(np.int32)

# 截掉埋点左侧的f

def mystr(s):

s = s.lstrip("f")

return s

csv\_file = open(r"C:\Users\14531\Desktop\testpoint.CSV") # 读埋点

csv\_reader\_lines = csv.reader(csv\_file)

date\_PyList = []

for one\_line in csv\_reader\_lines:

date\_PyList.append(one\_line[0])

date\_ndarray0 = list(map(mystr, date\_PyList))

fx = list(map(int, date\_ndarray0))

csv\_file.close()

end = np.zeros((17000, 123), dtype=np.int8)

k = 0

m = 52589302

indexx=[52589302]

for index, f in enumerate(fx):

temp = idx[index] # temp为i

end[k][f] += 1

if m != temp:

k += 1

m = temp

indexx.append(m)

if k>14998:

break

print(len(indexx))

answer=[[],[]]

results = exported\_pipeline.predict(end[:11523, 1:])

print(len(results))

answer[0]=list(indexx)

answer[1]=list(results)

answer=np.asfarray(answer)

print("开始存")

answer=np.transpose(np.array(answer))

with open(r"C:\Users\14531\Desktop\2t.csv", "w",newline="")as f:

writer = csv.writer(f)

for row in answer:

writer.writerow(row)