# 天池大数据竞赛重复购买预测问题的研究

## **学院：信息学院**

## **年级：2017级**

## **专业班级：计算机科学与技术2班**

**姓名：邓晓榕**

## 摘要

本次研究主要为了解决预测顾客重复购买同一商家商品概率的问题。针对这一问题，我们进行了数据预处理、特征工程处理、建立模型、检验模型效果、改进模型等步骤，最终成功解决了这一问题。

## 问题介绍

在特定日期（比如“黑色星期五”或者“双十一”），商家会进行大促销，以吸引新买家。但由于很多买家都不会复购，这对于商家会造成一定的损失。为提高投资回报率，商家需要判断出哪些顾客是较为可能复购的。我们的任务是预测指定商家的新顾客在6个月内再次从同一商家购买商品的概率。我们的训练集是一个包含大约20万用户信息的数据集，我们的测试集是一个包含商家ID和顾客ID的数据集。

## 模型设计

这个问题的数据预处理和特征工程部分已由我的队友完成。下面对这个问题进行建模。这个问题中顾客只有“会重复购买”和“不会重复购买”两种可能的状态，是一个典型的“二分类”问题，我们选用逻辑回归模型对它进行建模。

### 逻辑回归模型的基本原理

①逻辑回归原理：

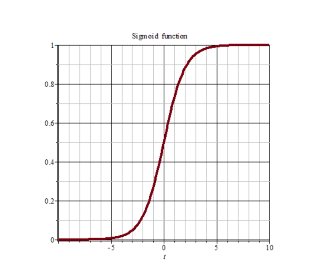


图 1

逻辑回归是用于处理分类问题的。对于线性回归，我们可以通过引入Sigmoid函数，将线性回归方程z变换为g(z)，并且令g(z)的值分布在(0,1)之间，且当g(z)接近0时样本的标签为类别0，当g(z)接近1时样本的标签为类别1，从而实现分类。逻辑回归的求解可以用梯度下降法求解。

②逻辑回归的损失函数：

逻辑回归的损失函数是由极大似然估计推导出来的，具体结果可以写作：



其中，表示求解出来的一组参数，m是样本个数，是样本i上真实的标签，是样本i上，基于参数计算出来的逻辑回归返回值，是样本各个特征的取值。

③正则化：

正则化是用来防止模型过拟合的过程。常用的有L1正则化和L2正则化两个选项，分别通过在损失函数后加上参数向量的L1范式和L2范式的倍数来实现。这个增加的范式，被称为“正则项”，也被称为“惩罚项”。损失函数改变，基于损失函数的最优化来求解的参数取值必然改变，我们以此来调节模型拟合的程度，其中L1范式表现为参数向量中的每个参数的绝对值之和，L2范式表现为参数向量中的每个参数的平方和的开方值。



### 

其中是我们之前提过的损失函数，C是用来控制正则化程度的超参数，n是方程中特征的总数，也是方程中参数的总数，j代表每个参数。在这里，j要大于等于1，是因为我们的参数向量中，第一个参数是，是我们的截距，它通常是不参与正则化的。

### sklearn中逻辑回归相关参数

1. max\_iter:最大迭代次数

在sklearn中我们设置参数max\_iter来代替步长，帮助我们控制模型的迭代速度并适时地让模型停下。max\_iter越大，代表步长越小，模型迭代时间越长，反之，则代表步长设置很大，模型迭代时间很短。

1. 选择求解器的参数solver

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **求解器** | **“liblinear”** | **“libfgs”** | **“newton-cg”** | **“sag”** | **“saga”** |
| 求解器  对应的  求解方  式 | 坐标下降  法 | 拟牛顿  法的一  种 | 牛顿法的一  种 | 随机平  均梯度  下降 | 随机平  均梯度  下降的  进化 |
| 支持的  惩罚项 | L1、L2 | L2 | L2 | L2 | L1、L2 |
| **支持的回归类型** | | | | | |
| MvM | 否 | 是 | 是 | 是 | 是 |
| OvR | 是 | 是 | 是 | 是 | 是 |
| 二分类 | 是 | 是 | 是 | 是 | 是 |
| **求解器的效果** | | | | | |
| 惩罚截  距 | 是 | 否 | 否 | 否 | 否 |
| 在大型  数据集  上更快 | 否 | 否 | 否 | 是 | 是 |
| 对未标  准化的  数据集  很有用 | 是 | 是 | 是 | 否 | 否 |

表 1

③正则化参数

正则化是用来防止模型过拟合的过程，可用penalty参数来选择使用L1正则化还是L2正则化。C是用来控制正则化程度的超参数。

L1和L2的区别：L1正则化会将参数压缩为0，L2正则化只会让参数尽量小，不会取到0。

|  |  |
| --- | --- |
| 参数 | 说明 |
| penalty | 可以输入”l1”或”l2”来指定使用哪一种正则化方式，不填写默认”l2”。注意：若选择”l1”正则化，参数solver仅能够使用求解方式”liblinear”和”saga”，若使用”l2”正则化，参数solver中所有的求解方式都可以使用。 |
| C | C是正则化强度的倒数，必须是一个大于0的浮点数，不填写默认1.0，即默认正则项与损失函数的比值是1:1.C越小，损失函数会越小，模型对损失函数的惩罚越重，正则化效力越强，参数会逐渐被压缩得越来越小。 |

表 2

### 建立逻辑回归模型

1.导入库

from sklearn.linear\_model import LogisticRegression as LR

#导入逻辑回归库

import numpy as np #导入numpy库（用于数值计算）

import matplotlib.pyplot as plt #导入用于绘图的库

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

#导入用于划分训练集、测试集的库

from sklearn.metrics import accuracy\_score

#导入用于计算分数的库

import scikitplot as skplt

#导入scikitplot库（用于绘图，如ROC曲线）

2.尝试训练模型

data=train

#本次实验的训练集，在前面数据预处理的代码中被赋给train，现在我们将它赋给data

x=data.iloc[:,data.columns!='label']

#数据集中所有列名不是”label”的列，构成特征矩阵x

y=data.iloc[:,data.columns=='label']

#数据集中列名为”label”的列是标签y

Xtrain,Xtest,Ytrain,Ytest=train\_test\_split(x,y,test\_size=0.3,

random\_state=420)

#划分测试集、训练集，训练集占70%，测试集占30%

lrl=LR(penalty='l2',solver='liblinear',C=0.5,max\_iter=100)

lrl=lrl.fit(x,y) #进行训练

lr.score(Xtest,Ytest) #得出分数（准确率：Accuracy）

此时，得到的分数为0.9349517584811703，在这里分数的评估指标是准确率（accuracy）.看起来这个分数还不错，下面看一下ROC曲线的评估结果。

3.绘制ROC曲线

vali\_proba\_df = pd.DataFrame(lr.predict\_proba(Xtest))

skplt.metrics.plot\_roc(Ytest, #真正的测试集标签

vali\_proba\_df, #预测的测试集标签

plot\_micro=False,

figsize=(6,6),

#设置图的大小，宽、高均为6

plot\_macro=False)

ROC曲线见附录图2.尽管本模型的accuracy不错，但它的ROC曲线却非常糟糕，AUC面积仅为0.51.于是我们要对模型进行调参。

4.调参

(1)绘制不同C值的学习曲线，尝试找到较好的C值

（学习曲线的横坐标为C值，纵坐标为准确率（Accuracy））

data=train

x=data.iloc[:,data.columns!='label']

y=data.iloc[:,data.columns=='label']

Xtrain,Xtest,Ytrain,Ytest=train\_test\_split(x,y,test\_size=0.3,

random\_state=420)

c\_1 = np.linspace(0.005,1,20)

#从0.005到1之间等距离选取20个C值

score = []

#建一个score的空列表

for i in c\_1:

lr = LR(solver='liblinear',C=i).fit(x,y) #进行训练

score.append(lr.score(Xtest,Ytest))

#将分数放入score列表中储存起来，以备画图

plt.figure() #画布

plt.plot(c\_1,score) #设置图的横坐标为C值，纵坐标为分数

plt.show() #显示图

c\_1[score.index(max(score))]

#求出当score取得最大值的时候C的值

C值的学习曲线见附录图3，可以计算出分数最高时的C值为0.21447368421052632.

#训练

data=train

x=data.iloc[:,data.columns!='label']

y=data.iloc[:,data.columns=='label']

Xtrain,Xtest,Ytrain,Ytest=train\_test\_split(x,y,test\_size=0.3,

random\_state=420)

lrl=LR(penalty='l2',solver='liblinear',C=0.2145,max\_iter=100)

lrl=lrl.fit(x,y)

lr.score(Xtest,Ytest)

#绘制ROC曲线

vali\_proba\_df = pd.DataFrame(lr.predict\_proba(Xtest))

skplt.metrics.plot\_roc(Ytest, vali\_proba\_df,

plot\_micro=False,figsize=(6,6),

plot\_macro=False)

根据前面计算出来的C值，绘制出来的ROC曲线见附录图4，可以看出AUC面积仍为0.51，效果仍不理想。

(2)修改正则化参数

data=train

x=data.iloc[:,data.columns!='label']

y=data.iloc[:,data.columns=='label']

Xtrain,Xtest,Ytrain,Ytest=train\_test\_split(x,y,test\_size=0.3,

random\_state=420)

lrl=LR(penalty='l1',solver='liblinear',C=0.5,max\_iter=100)

lrl=lrl.fit(x,y)

lr.score(Xtest,Ytest)

#绘制ROC曲线

vali\_proba\_df = pd.DataFrame(lr.predict\_proba(Xtest))

skplt.metrics.plot\_roc(Ytest,

vali\_proba\_df,

plot\_micro=False,

figsize=(6,6),

plot\_macro=False)

将正则化参数由l2修改为l1，得到的ROC曲线见附录图5，可以看到AUC面积仍为0.51，结果仍然不理想。

（3）修改solver值

data=train

x=data.iloc[:,data.columns!='label']

y=data.iloc[:,data.columns=='label']

Xtrain,Xtest,Ytrain,Ytest=train\_test\_split(x,y,test\_size=0.3,

random\_state=420)

lrl=LR(penalty='l2',solver='newton-cg',C=0.58,max\_iter=100)

lrl=lrl.fit(x,y)

lr.score(Xtest,Ytest)

#绘制ROC曲线

vali\_proba\_df = pd.DataFrame(lr.predict\_proba(Xtest))

skplt.metrics.plot\_roc(Ytest,

vali\_proba\_df,

plot\_micro=False,

figsize=(6,6),

plot\_macro=False)

修改solver为newton-cg，C值在做了多次调试后，找到了相对较好的值0.58，此时绘制出来的ROC曲线见附录图6.我们欣喜地发现，经过不断尝试后，AUC面积的值终于从0.51提高到了0.63.

## 测试结果

经过不断调整参数，本模型最终的测试结果如下：

①准确率（Accuracy）：0.93588546529723

②ROC曲线：见附录图5，AUC面积为0.63

## 结论

该模型能够对数据进行一定程度的准确预测，但模型的预测效果仍然不是非常理想。究其原因，可能有以下几方面：

①数据预处理和特征工程处理还存在一定的不完善之处。

②模型调参还没有找到最优的参数。

③逻辑回归模型本身对于该数据集的拟合效果就比较一般。

通过本次研究，我们可以得出逻辑回归模型的优点如下：

①形式简单，模型的可解释性非常好。从特征的权重可以看到不同的特征对最后结果的影响，某个特征的权重值比较高，那么这个特征最后对结果的影响会比较大。  
②模型效果不错。在工程上是可以接受的，如果特征工程做的好，效果不会太差，并且特征工程可以大家并行开发，大大加快开发的速度。  
③训练速度较快。分类的时候，计算量仅仅只和特征的数目相关。④资源占用小,尤其是内存。因为只需要存储各个维度的特征值。  
⑤方便输出结果调整。逻辑回归可以很方便的得到最后的分类结果，因为输出的是每个样本的概率分数，我们可以很容易的对这些概率分数进行划分阈值(大于某个阈值的是一类，小于某个阈值的是一类)。

通过本次研究，我们可以得出逻辑回归模型的缺点如下：

①准确率并不是很高。因为形式非常的简单(非常类似线性模型)，很难去拟合数据的真实分布。  
②很难处理数据不平衡的问题。

③处理非线性数据较麻烦。逻辑回归在不引入其他方法的情况下，只能处理线性可分的数据，或者进一步说，处理二分类的问题 。  
④逻辑回归本身无法筛选特征。有时候，我们会用gbdt来筛选特征，然后再上逻辑回归。

## 参考文献

[1]周志华.机器学习[M].北京：清华大学出版社，2016

## 附录

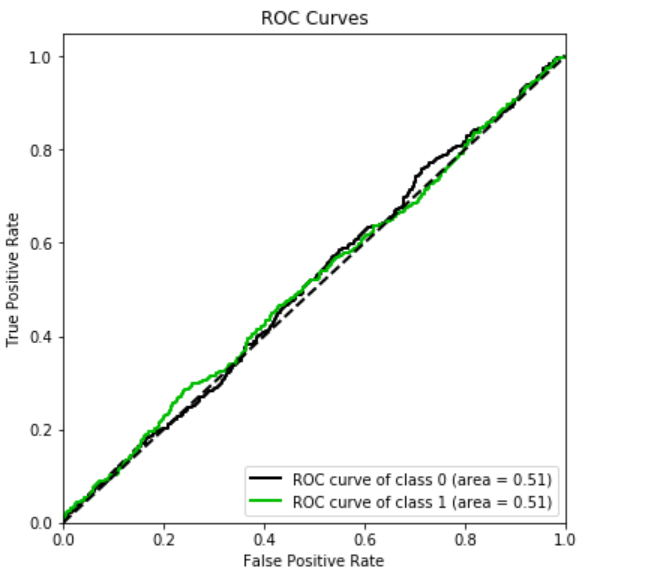


图 2

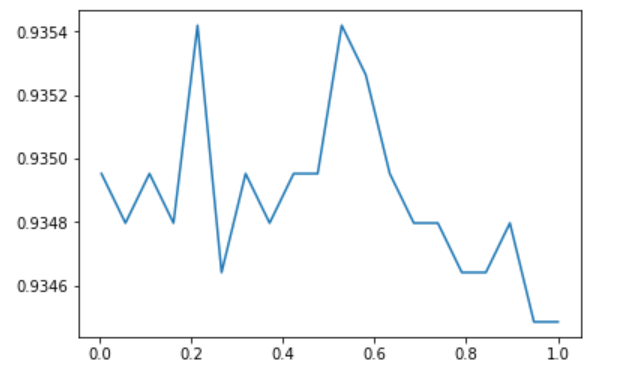


图 3

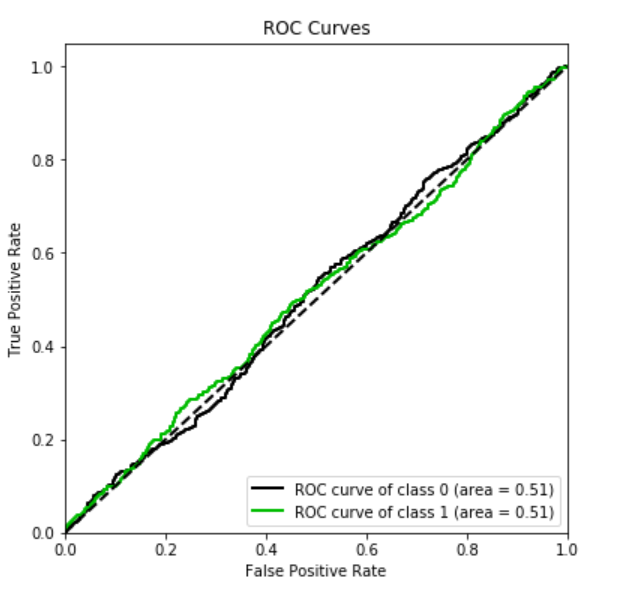


图 4

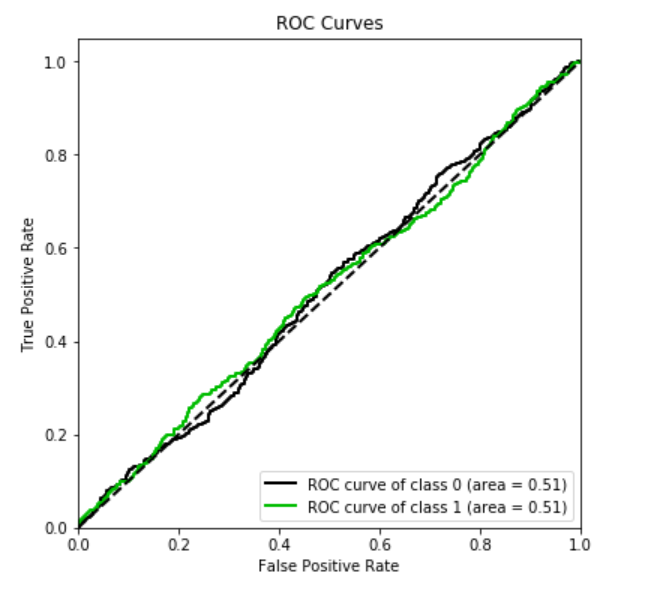


图 5

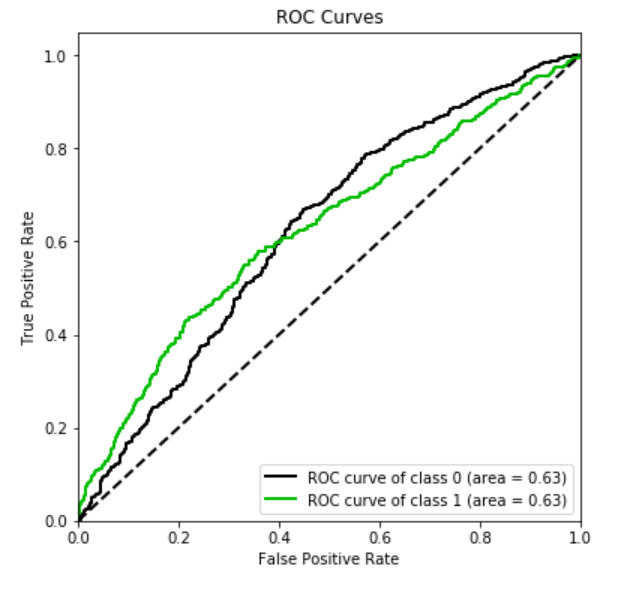


图 6