# 汽车保险预测技术报告

**方案一：Logistic回归**

* 工程文件说明

main.py：项目入口，原始数据读取、数据预处理、模型训练、模型评估、结果预测

train.py：搭建逻辑回归模型，梯度上升最小化误差，输出当前迭代score得分

test.py：计算F\_score得分，预测分类

utils.py：数据读取、预处理，交叉验证评估正则项参数，模型分布，下采样、过采样

data

| VI\_train.csv

| VI\_test.csv 原始数据

| train.csv

| predict.csv 预处理后数据

| submission.json 结果文件

* 数据预处理

通过pandas读取csv格式数据，利用dataframe支持的函数式编程可以方便的对原始数据进行查看和处理。这里以原始数据VI\_train.csv为例说明对数据的预处理。



图1 VI\_train.csv

首先将非数值型特征数值化，例如属性名为Gender的列，通过下列函数将其数值化，对于Vehicle\_Age一列，‘>2 Years’设置为1，‘1-2 Years’设置为0.5，‘<1 Years’设置为0。Vehicle\_Damage一列，‘Yes’设置为1，‘No’设置为0。

|  |
| --- |
| data["Gender"][data["Gender"] == 'Male'] = 0  data["Gender"][data["Gender"] == 'Female'] = 1 |

然后进行标准化，这里分为最小-最大规范化和零-均值规范化，例如属性名为Age的列，通过pandas函数式编程思想对其采用最小-最大规范化。对于Annual\_Premium一列，采用零-均值规范化，编写方法与下方类似。对于Policy\_Sales\_Channel一列，小于100的取0值，小于140取0.5，大于140取1。

|  |
| --- |
| f1 = lambda x: float(x - 20) / 65  data['Age'] = data['Age'].apply(f1) |

其余的数据清洗，将Driving\_License为0的数据去除，将Annual\_Premium大于164000的数据去除。最后数据特征选用constant(常数1)、Gender、Age、Previously\_Insured、Vehicle\_Age、Vehicle\_Damage、Annual\_Premium、Policy\_Sales\_Channel、Response。数据清洗及最终特征选择如图2所示。

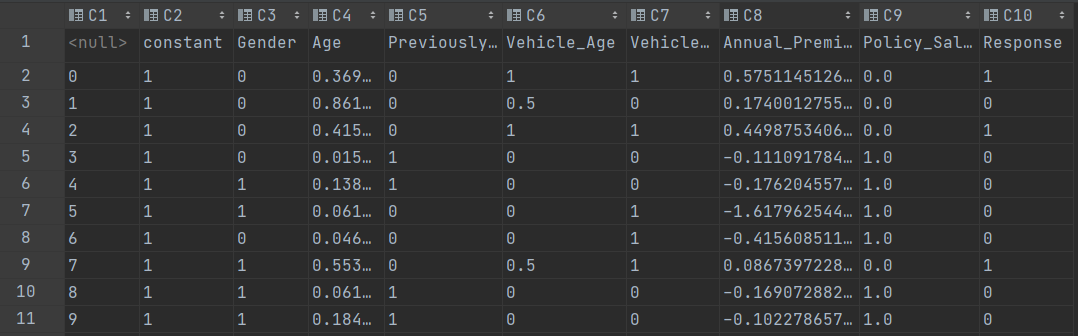


图2 train.csv

查看pyplot绘制数据分布，如图3所示，发现样本及其不均衡，因此考虑结合下采样和过采样对数据进一步处理。

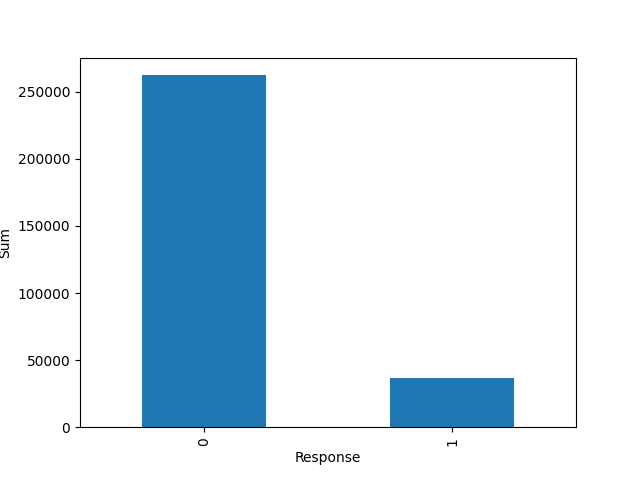


图3 原始数据阳/阴性分布

下采样，对于清洗之后的299231条数据，其中阴性样本个数为262409，阳性样本36822条，首先考虑使用下采样，采样方法是统计出数据阴性样本的索引值，然后在其中随机选取36822\*α条阴性样本，其中，α用于控制下采样程度，测试选取α为1时，F-score为0.3649，α为4时，F-score为0.3958，α为6时，F-score为0.3959。

上采样，选择imblearn.over\_sampling下的SMOTE样本生成策略，通过k近邻的思想生成阳性样本，通过设定参数k，生成k倍于原阳性样本个数的阳性数据，选择k=4，最终生成了409697条数据，其中阴性样本个数为262409，阳性样本147288条。

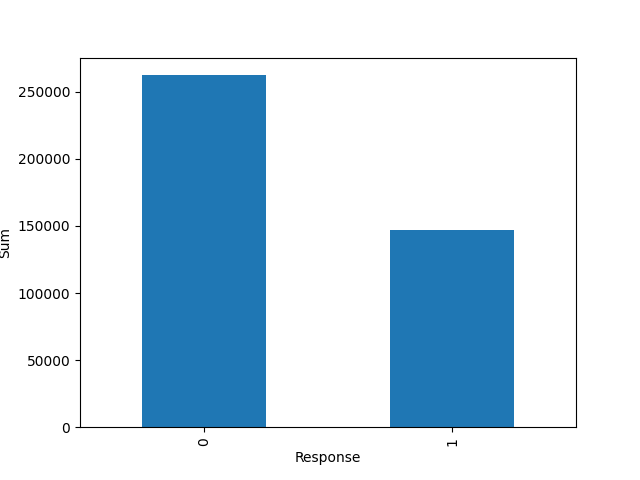


图4 过采样后的样本分布

* 模型原理

logistic回归模型，基于sigmoid函数作为输出，输出值限制在（0，1）上，有利于进行二分类模型的训练，sigmiod函数的输入记为z，其中z = w0x0+w1x1+w2x2+...+wnxn (这里0，1，2，...,n都代表下标系数)，简单写就是 z = wTx （T代表转置）。通过设定损失函数，算出误差，之后利用梯度上升最小化误差，提升逻辑回归预测概率的最大值，以得到最佳系数。之后，在预测阶段，便可根据该参数，算出sigmoid之后的值，并与设定的阈值比较，得到最终结果。

* 模型训练

利用梯度上升法的思想，结合随机选取数据，一次训练仅用一个样本点来更新回归系数,减少计算量。然而单纯使用该方法精度不高，相比较普通的梯度上升算法，每个数据被遍历到的次数减小到了1/iters，（iters为迭代次数）因此需要对该方法做适当改进。改进思想一是增加alpha动态减少的机制，这样做的原因是为了保证在多次迭代之后新数据仍然具有一定的影响。二是通过随机选取样本来更新回归系数，这种方法减少周期性的波动，在每轮迭代中，每处理完一个数据后，都需要将该值从列表中删掉，以保证其余数据能够被遍历到。

训练前通过train\_test\_split随机划分训练集和测试集，测试集占0.3，训练过程中，每轮迭代打印出当前F-score得分。训练中间过程如图所示。

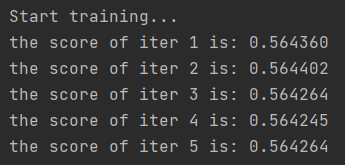


图5 训练中间过程

考虑增加l2正则项，通过KFlod设置交叉验证，来计算不同正则系数C下的recall值，C值的选取范围为[0.01,0.1,1,10,100]，计算过程如下图所示。最终判定1为较好的正则化系数。

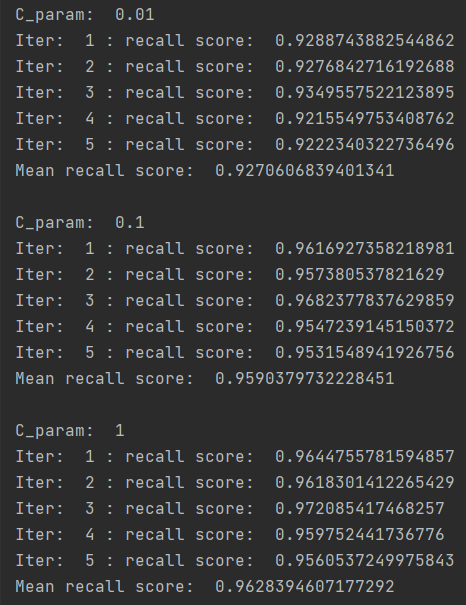
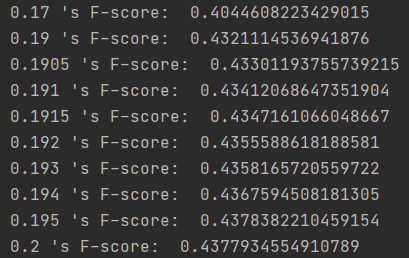


图6 不同正则项系数对应的recall

这里由于样本极度不均衡，因此判定阈值的设定也很重要，通过绘制混淆矩阵，查看计算不同阈值所对应的F-score，判定阈值范围为[0.17, 0.19, 0.1905, 0.191, 0.1915, 0.192, 0.193, 0.194, 0.195, 0.20,]，计算过程如图所示,最终选择0.1915作为判定阈值。



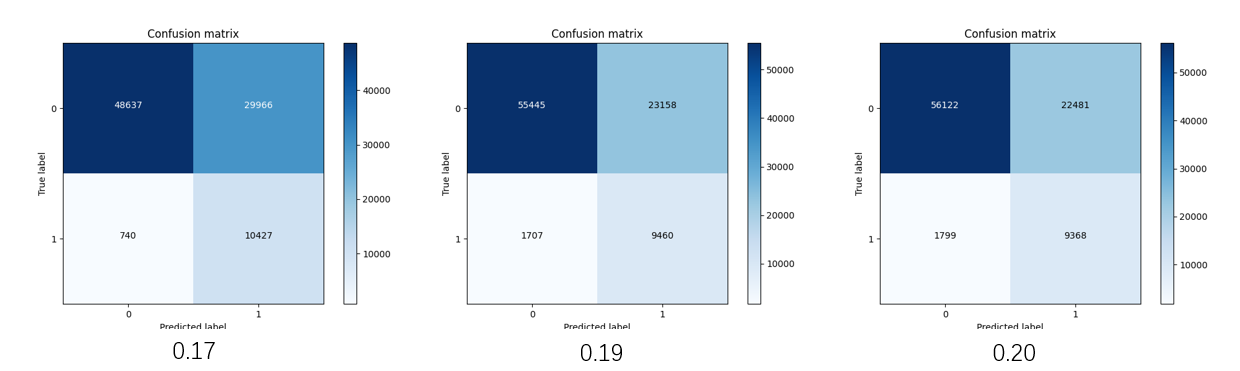


图7 不同输出阈值对应的F-score

* 排名截图

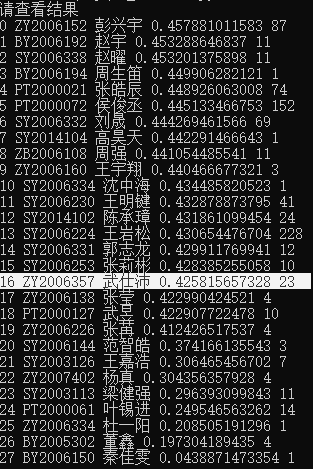


图8 排名截图

**方案二：神经网络**

* 工程文件说明

main.py：项目入口，原始数据读取、数据预处理、模型创建、模型训练、预测及输出

model.py：神经网络模型，参数初始化、前向传播、反向传播、参数更新、预测模块

utils.py：数据读取、预处理，交叉验证评估正则项参数，模型分布，下采样、过采样

data

| train.csv

| predict.csv 预处理后数据

| submission.json 结果文件

* 模型原理

使用numpy实现三层神经网络，模块主要包含init\_parameters、forward\_propagation、back\_propagation、update\_parameters四大模块。

init\_parameters：分别对每层参数进行初始化，从输入层开始，到最后一层隐藏层，权重参数W在0到0.01间随机，偏置参数b初始化为0。

forward\_propagation：从输入层到最后一层隐藏层激活函数采用relu或tanh激活函数（可通过参数选择），输出层激活函数采用sigmoid，前向传播过程中保存Zl（矩阵点积）和Al（激活值），方便反向传播计算梯度。

back\_propagation：输出层和隐藏层激活函数不同，需分别计算，对于输出层的sigmoid激活函数，求导公式为σ’(z) = σ(1-σ)。对于隐藏层，如果激活函数为tanh，求导公式为σ’(z)=1-(tanh)2，relu求导为σ’(z)=1（z>0）。

update\_parameters：通过梯度下降算法更新参数。

* 模型训练

模型搭建，各层节点个数为[8，6，4，1]，学习率设值0.05，激活函数采用relu，迭代次数为10000，每200轮输出当前损失值（二分类）。

数据划分，训练集70%，测试集30%。一开始选择教导较大学习率，导致后期损失函数始终降不下来，因此采取较小学习率的做法，综合更新速度和准确率，最终选择0.001。

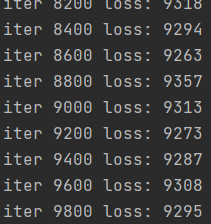


图9 较大学习率导致的梯度震荡

考虑采用L2正则化和dropout减少过拟合，dropout在每一次训练时，会随机消除一部分单元，使网络的输出不过分依赖于某一个节点，在采用dropout训练后，预测过程要对输出进行rescale。

激活函数的选择，采用relu作为隐藏层激活函数，该激活函数运算简单，学习速率快，同样保存了部分非线性表达能力，tanh输出结果更好，但训练速度相对较慢。

网络层数的选择上，越深层次的神经网络模型训练越慢，存在过拟合和局部最优解的情况，因此仅考虑3层网络模型。

* 最终结果排名

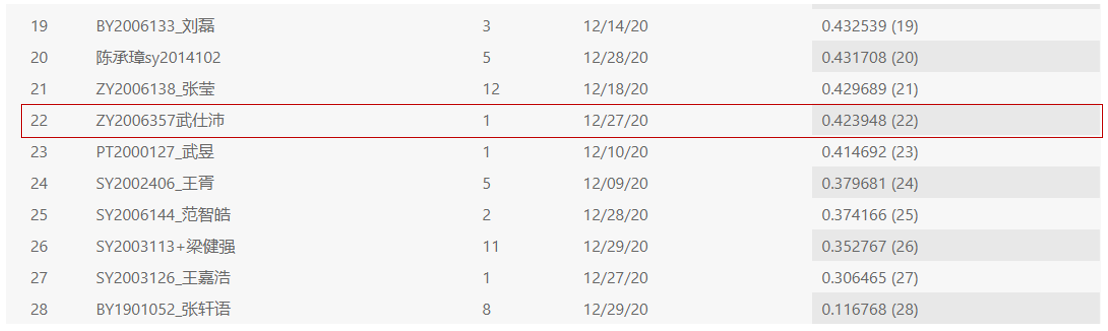


图10 最终排名截图