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。

	C3		C5	C6	C7	C8	C9	C10	C11	C12	C13
1	Gender	Age	Driving_…	Region_Code	Previousl…	Vehicle_…	Vehicle_…	Annual_…	Policy_…	Vintage	Response
2	Male	44	1	28.0	0	> 2 Years	Yes	40454.0	26.0	217	1
3	Male	76	1	3.0	0	1-2 Year	No	33536.0	26.0	183	0
4	Male	47	1	28.0	0	> 2 Years	Yes	38294.0	26.0	27	1
5	Male	21	1	11.0	1	< 1 Year	No	28619.0	152.0	203	0
6	Female	29	1	41.0	1	< 1 Year	No	27496.0	152.0	39	0
7	Female	24	1	33.0	0	< 1 Year	Yes	2630.0	160.0	176	0
8	Male	23	1	11.0	0	< 1 Year	Yes	23367.0	152.0	249	0
9	Female	56	1	28.0	0	1-2 Year	Yes	32031.0	26.0	72	1

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。

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	<null>	constant	Gender	Age	Previously…	Vehicle_Age	Vehicle…	Annual_Premi…	Policy_Sal…	Response
2	0	1	0	0.369…	0	1	1	0.5751145126…	0.0	1
3	1	1	0	0.861…	0	0.5	0	0.1740012755…	0.0	0
4	2	1	0	0.415…	0	1	1	0.4498753406…	0.0	1
5	3	1	0	0.015…	1	0	0	-0.111091784…	1.0	0
6	4	1	1	0.138…	1	0	0	-0.176204557…	1.0	0
7	5	1	1	0.061…	0	0	1	-1.617962544…	1.0	0
8	6	1	0	0.046…	0	0	1	-0.415608511…	1.0	0
9	7	1	1	0.553…	0	0.5	1	0.0867397228…	0.0	1
10	8	1	1	0.061…	1	0	0	-0.169072882…	1.0	0
11	9	1	1	0.184…	1	0	0	-0.102278657…	1.0	0

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。

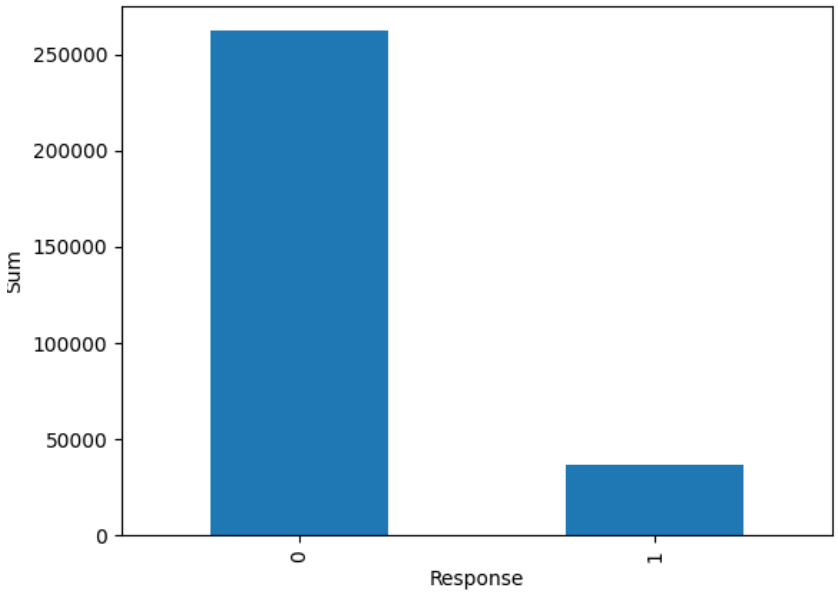


图 3 原始数据阳/阴性分布

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。

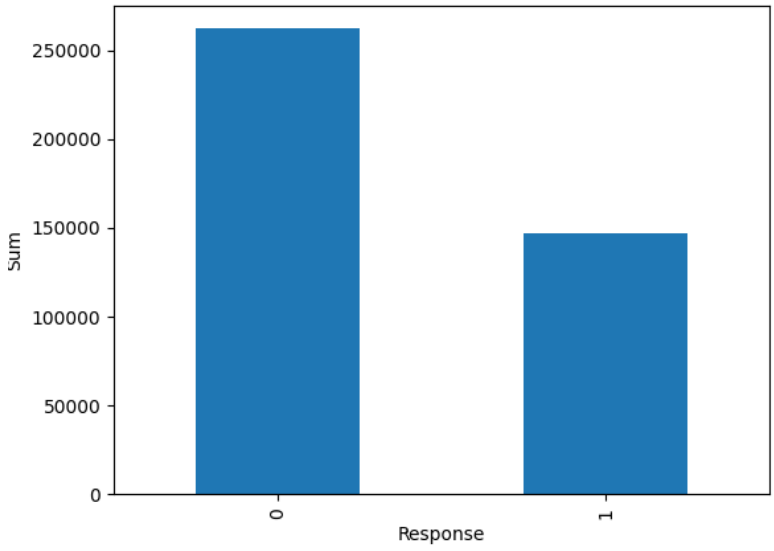


图 4 过采样后的样本分布

- 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。

```
Start training...
the score of iter 1 is: 0.564360
the score of iter 2 is: 0.564402
the score of iter 3 is: 0.564264
the score of iter 4 is: 0.564245
the score of iter 5 is: 0.564264
```

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

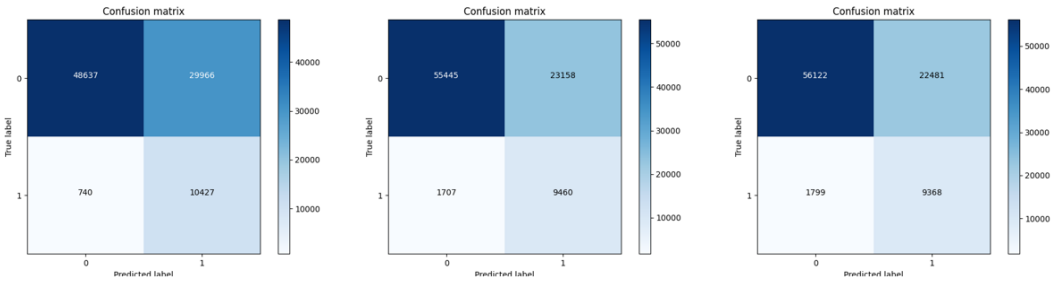
值，C 值的选取范围为[0.01,0.1,1,10,100]，计算过程如下图所示。最终判定 1 为较好的正则化系数。

```
C_param:  0.01
Iter:  1 : recall score:  0.9288743882544862
Iter:  2 : recall score:  0.9276842716192688
Iter:  3 : recall score:  0.9349557522123895
Iter:  4 : recall score:  0.9215549753408762
Iter:  5 : recall score:  0.9222340322736496
Mean recall score:  0.9270606839401341

C_param:  0.1
Iter:  1 : recall score:  0.9616927358218981
Iter:  2 : recall score:  0.957380537821629
Iter:  3 : recall score:  0.9682377837629859
Iter:  4 : recall score:  0.9547239145150372
Iter:  5 : recall score:  0.9531548941926756
Mean recall score:  0.9590379732228451

C_param:  1
Iter:  1 : recall score:  0.9644755781594857
Iter:  2 : recall score:  0.9618301412265429
Iter:  3 : recall score:  0.972085417468257
Iter:  4 : recall score:  0.959752441736776
Iter:  5 : recall score:  0.9560537249975843
Mean recall score:  0.9628394607177292
```

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。

```
0.17 's F-score:  0.4044608223429015
0.19 's F-score:  0.4321114536941876
0.1905 's F-score:  0.43301193755739215
0.191 's F-score:  0.43412068647351904
0.1915 's F-score:  0.4347161066048667
0.192 's F-score:  0.4355588618188581
0.193 's F-score:  0.4358165720559722
0.194 's F-score:  0.4367594508181305
0.195 's F-score:  0.4378382210459154
0.2 's F-score:  0.4377934554910789
```



0.17　　0.19　　0.20

图 7 不同输出阈值对应的 F-score

- 排名截图

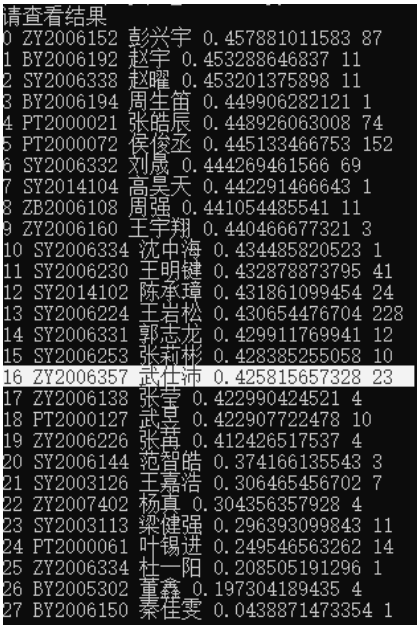


图 8 排名截图

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 Z1（矩阵点积）和 A1（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。

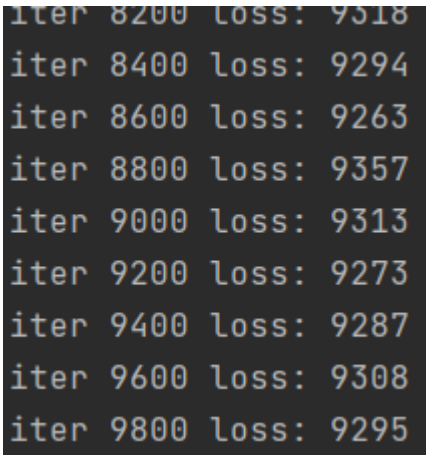


图 9 较大学习率导致的梯度震荡

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

19	BY2006133_刘磊	3	12/14/20	0.432539 (19)
20	陈承璋sy2014102	5	12/28/20	0.431708 (20)
21	ZY2006138_张莹	12	12/18/20	0.429689 (21)
22	ZY2006357武仕沛	1	12/27/20	0.423948 (22)
23	PT2000127_武昱	1	12/10/20	0.414692 (23)
24	SY2002406_王胥	5	12/09/20	0.379681 (24)
25	SY2006144_范智皓	2	12/28/20	0.374166 (25)
26	SY2003113+梁健强	11	12/29/20	0.352767 (26)
27	SY2003126_王嘉浩	1	12/27/20	0.306465 (27)
28	BY1901052_张轩语	8	12/29/20	0.116768 (28)

图 10 最终排名截图