法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：大数据分析挖掘
  - 新浪微博：ChinaHadoop

GBDT模型在流失预警中的应用

目录

GBDT模型简介

Gradient Boosting Decision Tree, 梯度提升树

特点

- 基于简单决策树的组合模型
- 沿着梯度下降的方向进行提升
- 只接受数值型连续变量

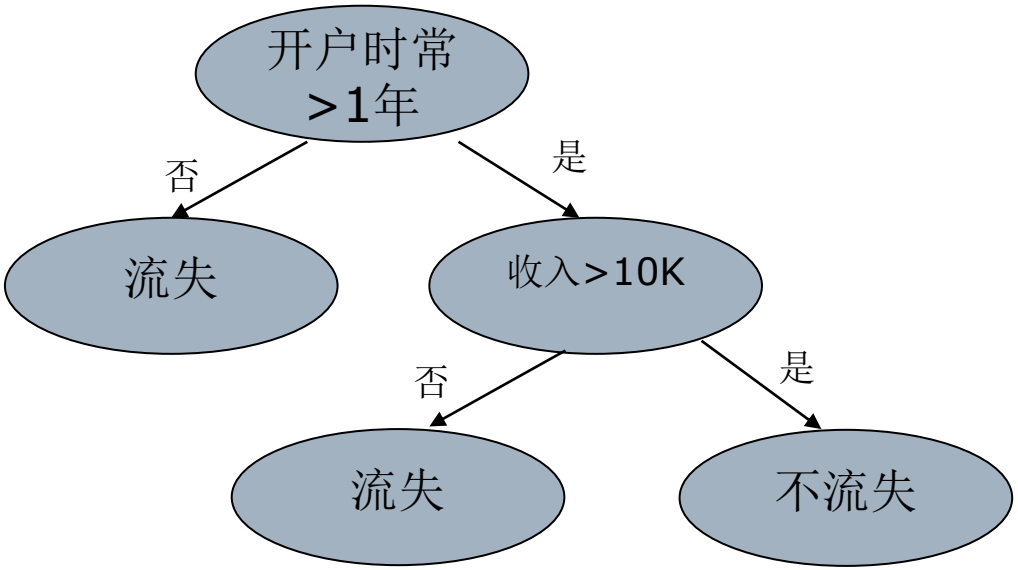
  需要做特征转化

优点

- 准确度高
- 不易过拟合

GBDT模型简介

决策树的复习



结构

- 根节点
- 中间节点

} 对特征的测试

- 叶节点　一组特征测试的结果

优点

- 可以分类和回归
- 解释性强
- 允许变量的交互作用
- 对离群值、缺失值、共线性不敏感

缺点

- 准确度不够高
- 易过拟合
- 运算量大

GBDT模型简介

组合模型

建立若干个简单的分类/回归树，组合在一起

Bagging

从同一样本、同一指标集里抽样，每次抽样都生成一棵简单树，可以并行建立

Boosting

模型建立有先后顺序，后一个模型是改进对前一个模型分类错误的结果赋予更大的权重

Stacking

模型建立有先后顺序，前一个模型的输出是后一个模型的输入

GBDT模型简介

GBDT模型的原理(以分类树为例)

结构

用$F(x)=\sum_k^K f_k(x)$来逼近y，y是二分类标签，K是分类树个数

损失函数(Loss Function)

- 第k步累计函数的损失=加上第k棵树后的精度损失(Training Loss)+加上第k棵树后的复杂度惩罚(Penalty on Complexity)
- 待求变量：第k棵树
- 目的：让第k步累计函数的损失最小(梯度法结合泰勒展式)
- 结束：将第k棵树加到之前的模型中

分类问题中常用的Training Loss

$$l(y,\hat{y})=yln\left(1+e^{-\hat{y}}\right)+(1-y)\ln(1+e^{y})$$

GBDT模型的参数

GBDT框架的参数

n_estimators: 弱在此处键入公式。分类树的个数，即K

learning_rate: 即每个弱学习器的权重缩减系数ν，也称作步长。较小的ν意味着需要更多的弱学习器的迭代次数。参数n_estimators和learning_rate要一起调参。可以从一个小一点的ν开始调参，默认是1

Subsample: (不放回)抽样率，推荐在[0.5, 0.8]之间，默认是1.0，即不使用子采样

init: 即初始化的时候的弱学习器，一般用在对数据有先验知识，或者之前做过一些拟合的时候

loss: 即GBDT算法中的损失函数

GBDT模型的参数(续)

弱分类树的参数

max_features: 划分时考虑的最大特征数

max_depth: 决策树最大深度

min_samples_split: 内部节点再划分所需最小样本数。默认是2.如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值

min_samples_leaf: 叶子节点最少样本数

min_weight_fraction_leaf：叶子节点最小的样本权重。默认是0，就是不考虑权重问题。一般来说，如果我们有较多样本有缺失值，或者分类树样本的分布类别偏差很大，就会引入样本权重，这时我们就要注意这个值了

max_leaf_nodes: 最大叶子节点数，通过限制最大叶子节点数，可以防止过拟合

min_impurity_split: 节点划分最小不纯度

目录

分类器性能指标简介

分类器性能指标

预测

事实		真	假
	真	TP	FN
	假	FP	TN

表示分类正确：

True Positive：本来是正样例，分类成正样例。

True Negative：本来是负样例，分类成负样例。

表示分类错误：

False Positive ：本来是负样例，分类成正样例，通常叫误报。

False Negative：本来是正样例，分类成负样例，通常叫漏报

分类器性能指标简介

分类器性能指标(续)

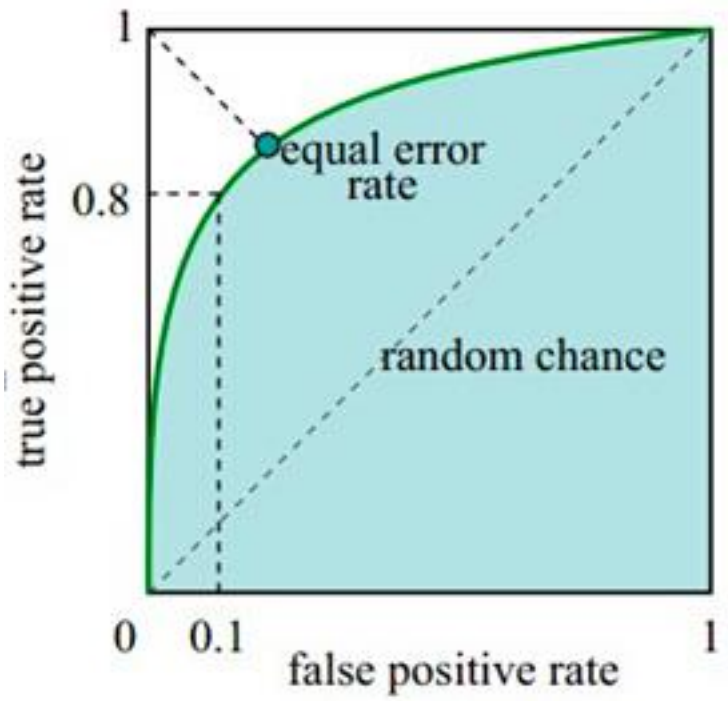
- 准确度 $\text{score} = \frac{TP+TN}{TP+TN+FP+FN}$
- 真正类率(True Postive Rate)TPR: **TP/(TP+FN)**,代表分类器预测的**正类中**实际正实例占所有正实例的比例。Sensitivity
- 负正类率(False Postive Rate)FPR: **FP/(FP+TN)**，代表分类器预测的**正类中**实际负实例占所有负实例的比例。1-Specificity
- 真负类率(True Negative Rate)TNR: **TN/(FP+TN)**,代表分类器预测的**负类中**实际负实例占所有负实例的比例，**TNR=1-FPR**。Specificity

分类器给出针对每个实例为正类的概率，那么通过设定一个阈值如0.6，概率大于等于0.6的为正类，小于0.6的为负类。对应的就可以算出一组(FPR,TPR),在平面中得到对应坐标点。随着阈值的逐渐减小，越来越多的实例被划分为正类，但是这些正类中同样也掺杂着真正的负实例，即TPR和FPR会同时增大。阈值最大时，对应坐标点为(0,0),阈值最小时，对应坐标点(1,1)。

分类器性能指标简介

分类器性能指标(续)

AUC是图中曲线下方的面积，值越大，分类效果越佳



目录

GBDT在流失预警模型中的应用

如何调参

- 训练集(training)　用于模型开发的一部分数据
- 测试集(testing)　用于测试模型的一部分数据
- Training和testing通常服从同一个分布

  如果不服从同一个分布？迁移学习

首先看下训练集中两种种类的分布

```
>>> y_train.value_counts()
0    7741
1     879
Name: CHURN_CUST_IND, dtype: int64
```

流失率：11.36%

GBDT在流失预警模型中的应用

如何调参

使用默认参数，在训练集上

Score = 92.75% ✔ AUC = 92.37% ✔

在测试集上

Score = 91.18% ✔ AUC = 85.40% ✔

都还不错。但是能不能更好点？

GBDT在流失预警模型中的应用

如何调参(续)

首先我们从步长(learning rate)和迭代次数(n_estimators)入手。一般来说,开始选择一个较小的步长来网格搜索最好的迭代次数。这里，我们将步长初始值设置为0.1，迭代次数的搜索范围是20～80

```
mean:  0.84439,  std:  0.01532,  params:  {'n_estimators':  20},
mean:  0.84840,  std:  0.01378,  params:  {'n_estimators':  30},
mean:  0.84898,  std:  0.01428,  params:  {'n_estimators':  40},
mean:  0.85013,  std:  0.01366,  params:  {'n_estimators':  50},
mean:  0.85065,  std:  0.01354,  params:  {'n_estimators':  60},
mean:  0.85105,  std:  0.01338,  params:  {'n_estimators':  70},
mean:  0.85059,  std:  0.01402,  params:  {'n_estimators':  80}
```

最好的迭代次数是70，对应的score是85.10%。

好像比默认参数的效果差。。。。

GBDT在流失预警模型中的应用

如何调参(续)

找到了一个合适的迭代次数，现在我们开始对决策树进行调参。首先我们对决策树最大深度max_depth和内部节点再划分所需最小样本数min_samples_split进行网格搜索。搜索的范围分别是3～13和 100～800

```
mean:  0.84762,  std:  0.01205,  params:  {'min_samples_spl 100,  'max_depth'3},
mean:  0.84956,  std:  0.01103,  params:  {'min_samples_spl 300,  'max_depth'3},
mean:  0.84934,  std:  0.01107,  params:  {'min_samples_spl 500,  'max_depth'3},
mean:  0.84836,  std:  0.01149,  params:  {'min_samples_spl 700,  'max_depth'3},
mean:  0.85164,  std:  0.01137,  params:  {'min_samples_spl 100,  'max_depth'5},
mean:  0.85311,  std:  0.01087,  params:  {'min_samples_spl 300,  'max_depth'5},
mean:  0.85070,  std:  0.01327,  params:  {'min_samples_spl 500,  'max_depth'5},
mean:  0.85134,  std:  0.01252,  params:  {'min_samples_spl 700,  'max_depth'5},
mean:  0.85286,  std:  0.01418,  params:  {'min_samples_spl 100,  'max_depth'7},
mean:  0.85052,  std:  0.01113,  params:  {'min_samples_spl 300,  'max_depth'7},
                                 .
                                 .
                                 .
mean:  0.85090,  std:  0.01226,  params:  {'min_samples_spl 700,  'max_depth'13}
```

最佳的最大深度和最小样本数分别是9和500，对应的score是85.36%

GBDT在流失预警模型中的应用

如何调参(续)

由于决策树深度9是一个比较合理的值，我们把它定下来，对于内部节点再划分所需最小样本数min_samples_split，我们暂时不能一起定下来，因为这个还和决策树其他的参数存在关联。下面我们再对内部节点再划分所需最小样本数min_samples_split和叶子节点最少样本数min_samples_leaf一起调参。调整范围分别是400～1000，以及60～100。

```
mean:  0.85331,  std:  0.01126,  params:  {'min_samples_spl:400,  'min_samples_leaf':  60},
mean:  0.85011,  std:  0.01169,  params:  {'min_samples_spl:500,  'min_samples_leaf':  60},
mean:  0.85120,  std:  0.01273,  params:  {'min_samples_spl:600,  'min_samples_leaf':  60},
mean:  0.85324,  std:  0.01217,  params:  {'min_samples_spl:700,  'min_samples_leaf':  60},
mean:  0.85202,  std:  0.01209,  params:  {'min_samples_spl:800,  'min_samples_leaf':  60},
mean:  0.85326,  std:  0.01313,  params:  {'min_samples_spl:900,  'min_samples_leaf':  60},
mean:  0.85484,  std:  0.01127,  params:  {'min_samples_spl:400,  'min_samples_leaf':  70},
                                 .
                                 .
                                 .
mean:  0.85205,  std:  0.01117,  params:  {'min_samples_spl:900,  'min_samples_leaf':  100}
```

最佳的最小样本数和叶节点最小样本数分别是500和70，对应的score是85.54%

GBDT在流失预警模型中的应用

如何调参(续)

调了这么多参数了，终于可以都放到GBDT类里面去看看效果了。现在我们用新参数拟合数据，在训练集上的表现：

Score = 92.76% ✔ AUC = 94.84% ✔

在测试集上

Score = 91.17% ✔ AUC = 85.36% ✔

竟然还不如默认参数的效果

主要原理是我们使用了0.8的子采样，20%的数据没有参与拟合。

GBDT在流失预警模型中的应用

如何调参(续)

现在我们再对最大特征数max_features进行网格搜索，范围从5 到25，最佳值是25.

BUT！

对于边界值，通常还要再放大范围。我们将范围扩大到30，最佳值是28.

再对子采样的比例进行网格搜索，范围从0.6到0.9， 最佳值是0.8.

现在我们基本已经得到我们所有调优的参数结果了。这时我们可以减半步长，最大迭代次数加倍来增加我们模型的泛化能力。再次拟合我们的模型，得到的最优步长是0.05,最大迭代次数是1000，在训练集和测试集上的表现是：

训练集

Score = 91.73% ✔ AUC = 90.31% ✔

测试集

Score = 91.19% ✔ AUC = 85.61% ✔

GBDT在流失预警模型中的应用

变量重要性

和随机森林一样，GBDT也可以给出特征重要性。

疑问

- 问题答疑： http://www.xxwenda.com/
  - 可邀请老师或者其他人回答问题

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop