题 目 基于传统机器学习与深度学习对信用卡欺诈交易的精准识别

关 键 词 强分类器，SMOTE算法，Adboost算法，AutoEncoder模型

摘 要：

近年来，随着信用卡的发行量的快速增长，未偿信贷总额也随着增加，信用卡风险形势不容小觑。本文利用“好贷网”提供的信用卡消费数据，通过传统机器学习和深度机器学习两种方法，设计出两种各有优势的的算法在新的信用卡消费数据产生时，进行对欺诈交易的精准识别，并在消费过程结束前进行有效的预警。

首先我们对数据进行预处理，针对在数据集中欺诈案例所占比例甚小，我们使用**下采样**与**过采样**对数据集进行处理，接着我们对与这两类进行分析，通过描述统计发现对的分类几乎没有影响，我们予以删除，同时对列进行了数据标准化。

**针对机器学习方法**（模型一），我们基于**单层决策树分类器**的**集成学习模型**建立了一个框架算法。我们将处理好的数据集中随机选取作为训练集，作为测试集，针对训练数据利用单层决策树算法建立了个**弱分类器**，通过迭代算法进行自适应参数调整学习，基于我们定义的各项指标，其中过采样数据集测试数据结果为召回率，精准率，为，误判率为 ；下采样数据集测试数据结果为召回率，精准率，为，误判率为，因此我们的模型具有强欺诈识别能力，且有较高的参考价值以及意义。

**针对深度学习方法**（模型二），我们基于与的开源顶层框架，搭建了模型。我们同样将数据集拆分为训练数据与测试数据，通过建立29个神经元作为输入层，使用层隐藏层作为编码器和解码器，并且采用****算法对其参数进行迭代更新，同时使用**L1范式正则化**帮助模型达到拟合能力和泛化能力的均衡，基于我们定义的各项指标，其中下采样数据集测试数据结果为为，分类结果准确率达 ；过采样数据集测试数据结果为为，分类结果准确率达 ，同样是参考意义较大的优的分类鉴别器。

最后，针对模型二中的主观赋值，我们进行了**鲁棒性检验**，结果显示模型二具有较强的鲁棒性，同时，在文末我们分析了两种方法对信用卡欺诈交易的精准识别的**优缺点**，并对模型的**改进与推广**提出了建议。

**Abstract**

In recent years, with the rapid growth of credit card issuance, the total amount of outstanding credit is also increasing. Credit card risk situation should not be underestimated. In this paper, we use the credit card consumption data provided by the "HaoDai net" to design the model. Through the two methods of traditional machine learning and deep machine learning, we design two algorithms that have their own advantages. When new credit card consumption comes into being, the program accurately identifies fraudulent transactions and conducts effective early warning before the end of the consumption process.

First we deal with the data. Because the proportion of fraud cases in data set is very small, we use **the down sampling** and **oversampling** to process the data sets. Then we analyze the two classes of Time and Amount. By describing the statistics, we find that the Time field has almost no effect on the Class field. We delete it and standardize the data of the Amount column.

For **Machine\_learning** (model 1), we build **a framework algorithm** based on **ensemble learning model** of **single level decision tree classifier**. We extract 70% of processed data sets as training data and 30% as test data. Based on the indicators we define, the results of the over sampling data set test data are recall **0.959**, the precision rate **0.987**, the F-measure **0.973**, and the error rate **2.65%**. The result of the test data of the down set data set is recall rate **0.936**, precision rate **0.978**, **F-measure 0.957**, and **erroneous judgement rate 4.13%**. Therefore, our model has strong fraud recognition ability, and has high reference value and significance.

For **Deep\_learning** (model two), we build the **Autoencoder** **model** based on the open source top-level framework of Tensorflow and keras. The data set is divided into training data and test data. By Building 29 neurons as the input layer, the 4 layer hidden layer is used as the encoder and decoder, and the algorithm is used to update the parameters of the data. At the same time, **the L1 canonical regularization** help model is used to achieve the balance of the fitting ability and generalization ability. Based on all the indicators we define, the results of the lower sample data set test data AUC are **0.944**, the accuracy of the classification results is **94.482%**; the result of the over sampling data set test data is **0.965**, and the accuracy of the classification results is **98.7396%**.

Finally, in view of the subjective assignment in model 2, we carry out the robustness test. The results show that model 2 has **strong robustness**. At the same time, we analyze the advantages and disadvantages of the two methods for the accurate identification of credit card fraud transactions, and put forward some suggestions for the improvement and promotion of the model.

**目录**

[一、 问题重述 6](#_Toc515894314)

[1.1 问题背景（行业现状） 6](#_Toc515894315)

[1.2 问题重述 6](#_Toc515894316)

[二、 分析方法与过程 6](#_Toc515894317)

[2.1 问题分析 6](#_Toc515894319)

[2.2 模型一算法流程 6](#_Toc515894320)

[2.3 模型二步骤算法流程 7](#_Toc515894321)

[三、 模型假设 7](#_Toc515894322)

[四、 符号说明 7](#_Toc515894323)

[五、 数据预处理 8](#_Toc515894324)

[5.1 样本选取与数据筛选 8](#_Toc515894330)

[5.1.1 样本选取 8](#_Toc515894331)

[5.1.2 数据筛选 8](#_Toc515894332)

[六、 模型的建立与求解 14](#_Toc515894333)

[6.1 模型一：基于单层决策树的集成学习模型的建立 14](#_Toc515894335)

[6.2 模型一求解 16](#_Toc515894336)

[6.2.1 实验数据预处理 16](#_Toc515894339)

[6.2.2 定义指标对模型的分类能力进行表达 16](#_Toc515894340)

[6.2.3 对弱分类器进行迭代 17](#_Toc515894341)

[6.3 模型二：基于的的深度学习模型 20](#_Toc515894342)

[6.4 模型二求解 21](#_Toc515894343)

[6.4.1 数据预处理 21](#_Toc515894346)

[6.4.2 搭建模型 21](#_Toc515894347)

[6.4.3 加载训练模型对测试数据进行识别 21](#_Toc515894348)

[七、 鲁棒性检验 26](#_Toc515894349)

[八、 模型评价 27](#_Toc515894350)

[8.1 模型一与模型二的比较 27](#_Toc515894353)

[8.2 模型的优、缺点 28](#_Toc515894354)

[8.3 模型的改进和推广 28](#_Toc515894355)

[8.2.1 模型一改进与推广 28](#_Toc515894358)

[8.2.2 模型二改进与推广 28](#_Toc515894359)

[九、 参考文献 29](#_Toc515894360)

[十、 附录 30](#_Toc515894361)

# 问题重述

## 问题背景

信用卡是一种非现金交易付款的方式，是简单的信贷服务。它是由银行或信用卡公司依照用户的信用度与财力发给持卡人，持卡人持信用卡消费时无须支付现金，待账单日时再进行还款。 信用卡的出现使人们的消费往来更加的方便、简洁。虽然信用卡的面世极大程度的优化人们日常支付消费模式，但由于它自身的技术、管控等方面存在缺陷,成为犯罪分子有机可乘的对象。近年来关于信用卡的犯罪呈现上升趋势、犯罪手法也层出不穷，主要表现为: 冒用他人信用卡诈骗和恶意透支诈骗、伪造信用卡诈骗、使用作废信用卡诈骗[1]。

据报道，我国信用卡累计发卡量2016年达4.65亿张，到2017年第三季度末信用卡授信总额为11.91万亿元，应偿信贷余额为5.17万亿元，信用卡逾期半年未偿信贷总额662.71亿元，故而信用卡风险形势不容小觑。因而有必要在新的信用卡消费数据产生时，通过算法的识别，使得我们在消费过程结束前进行有效的预警。

## 问题重述

数据分析公司希望得到一个有效的算法，使用这些已知字段，对信用卡是否欺诈进行尽可能正确的分类。在新的信用卡消费数据产生时，通过算法的识别，我们可以在消费过程结束前进行有效的预警。我们旨在采用数据挖掘技术能够快速、高效地检测和预防欺诈行为的发生，并且考虑数据集中欺诈案例所占比例甚小的情况，最终通过算法有效、精准识别来降低信用卡欺诈风险，维护正常的金融秩序, 促使银行信用卡业务的健康发展。

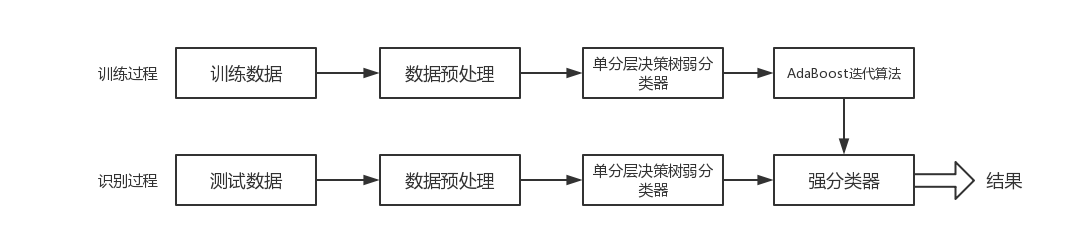
# 分析方法与过程



## 问题分析

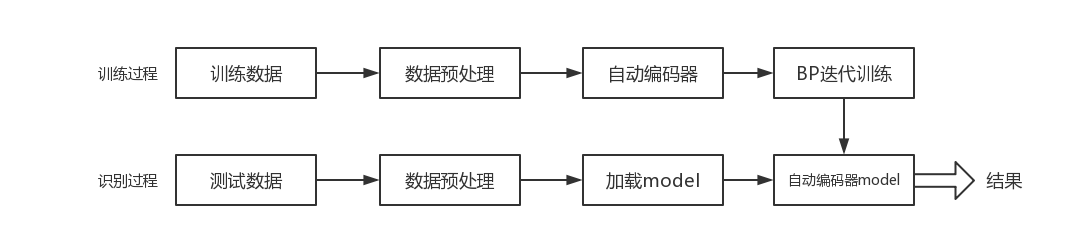
数据集是由28个字段经过PCA提取的没有具体含义的主成分，以及具有具体含义、和三个字段。在分析数据时发现，数据集中欺诈交易样本数据即为1的数目过少。因此，我们针对在数据集中欺诈案例所占比例甚小等问题，在数据预处理时对以上几种存在影响结果准确性的因素进行重新处理，得到现有的数据集。为了给与出题企业更好的选择，我们设计基于传统机器学习（模型一）与深度机器学习（模型二）两种算法对数据进行分类判断。

## 模型一算法流程



模型一算法流程图

## 模型二步骤算法流程



模型二算法流程图

# 模型假设

1. 假设所提供的数据集是正确收集而来，并非虚假的人工数据。
2. 假设指标字段 到 为有意义的指标。这是合理的，因为事实上主成分提取的指标是综合其对应的全部指标，其具体含义可视为代表全部指标的模拟趋势。
3. 假设在过采样中使用的算法中随机合理，不会产生模型过拟合的问题。
4. 假设每个由单层决策树构成的弱分类器分类能力大于先验假设。
5. 假设模型使用的训练误差就是真正的训练误差。
6. 假设自动编码器在同意阈值下对测试数据的输出结果均一致。

# 符号说明

|  |  |
| --- | --- |
| 符号 | 说明 |
|  | 训练样本集的权值发布 |
|  | 每个训练样本的权值大小 |
|  | 弱分类器 |
|  | 基本分类器 |
|  | 最终强分类器 |
|  | 误差率 |
|  | 弱分类器的权重 |
|  | 迭代训练传播次数 |
|  |  |
|  |  |
|  | 标准差 |
|  | 最小值 |
|  | 最大值 |

# 数据预处理



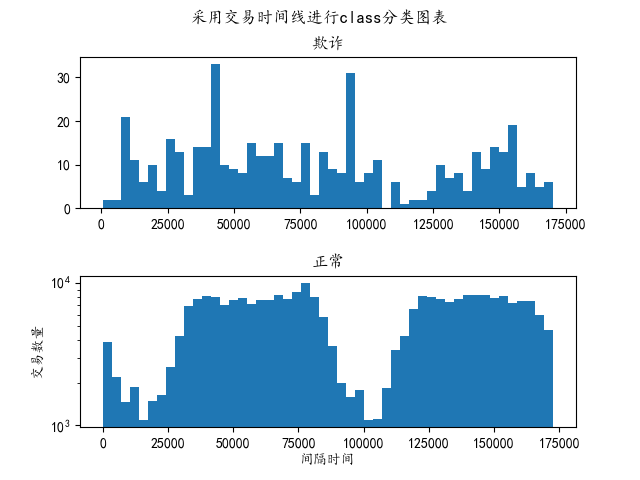
## 样本选取与数据筛选

### **样本选取**

本文数据均由好贷杯组委会提供，数据集是284785条信用卡消费记录。数据共分 31 个字段，为从数据集的第一条记录开始计算的时间，单位为秒。为刷卡消费的金额，为是否欺诈，在欺诈的情况下其值为 1，否则为0，其中值为1记录共有483条，值为0记录共有284302条。字段 到 是原数据集经过主成分分析提取的28 个主成分，且 28 个字段均无详细说明。

### **数据筛选**

1. **时间线判断**

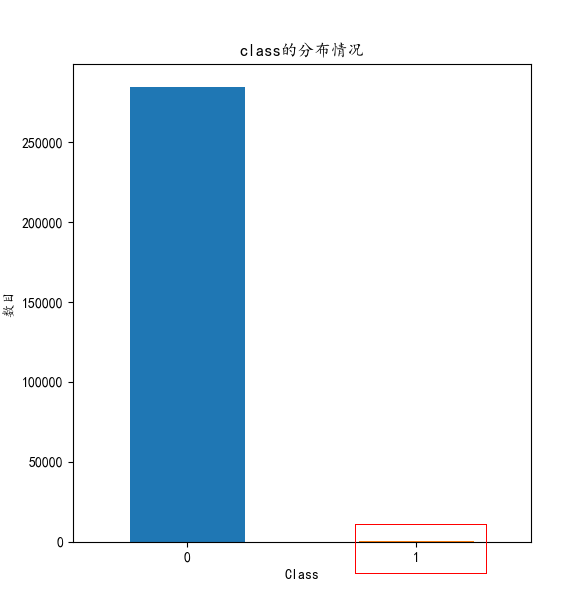


基于字段信用卡欺诈、正常分布对比图

基于时间线分析，我们通过对原数据集中字段做出简单判断处理，以交易时间为横坐标，交易数量为纵坐标得出结果如图3。根据图3结果显示得出信用卡欺诈、正常消费的时间并发趋势**大致相同**，故而通过时间字段判断是否存在消费欺诈行为的意义并不大，并且其最大最小值标准化差距过大，因此，我们在数据集中剔除了该指标。

1. **关于中0，1二类分布不平衡的处理**

基于字段分析，通过对原数据集统计，其中值为1记录共有483条，值为0记录共有284302条结果如图4。



分布情况

不难发现欺诈案例所占比例甚小。因此中0，1二类分布极度不平衡，可能会导致算法出现巨大隐患如下：

1. 从模型的训练过程来看：如果某类的样本数量很少，那么这个类别所提供的“信息”就太少,即使用经验风险（模型在训练集上的平均损失）最小化作为模型的学习准则，因此容易导致模型没有学习到如何去判别出少数类。
2. 从模型的预测过程来看：数据挖掘模型一般是出于最大后验概率决策的角度考虑的，这意味着当模型估计的样本属于正类的后验概率要大于样本属于负类的后验概率时就将样本判为正类。但实际上，这个后验概率的估计值由于数据极度分布不平衡导致无法准确评估，当一个样本的预测几率大于观测几率时，就应该将样本判断为样本较多的类。

基于以上问题，我们提出了两种解决方案：

1. **下采样**

根据统计我们知道，该数据集中为0的交易信息数据共有284302条，为1的交易信息数据共有483条，我们根据下采样定义，抽取为1的交易信息数据共有483条，随机抽取为0的交易信息数据483条。然后打乱顺序，制作成数据集**[2]。**

1. **过采样**

，合成少数类过采样技术．它是基于随机过采样算法的一种改进方案，由于随机过采样采取简单复制样本的策略来增加少数类样本，这样容易产生模型过拟合的问题，即使得模型学习到的信息过于特别()而不够泛化()，算法[3]的基本思想是对少数类样本进行分析并根据少数类样本人工合成新样本添加到数据集中，具体如下图所示，算法流程如下。

1. 对于少数类中每一个样本，以欧氏距离为标准计算它到少数类样本集中所有样本的距离，得到其近邻。
2. 根据样本不平衡比例设置一个采样比例以确定采样倍率，对于每一个少数类样本，从其近邻中随机选择若干个样本，假设选择的近邻为。
3. 对于每一个随机选出的近邻，分别与原样本按照如下的公式构建新的样本。

其数学模型如下：

(1)

因此，我们抽取483条为1的交易信息数据作为训练数据，通过上述算法生成284302条为1的数据，并且随机打乱交易信息序列，制作成数据集。

1. **数据标准化处理**

基于字段分析，由于指标携带量纲指标，为了方便模型训练，避免对准确率产生很大的影响，因此通过数据标准化处理即将数据特征缩放至某一范围，调用r[4]将数据归一化至区间内统计其概率分布。

(2)

(3)

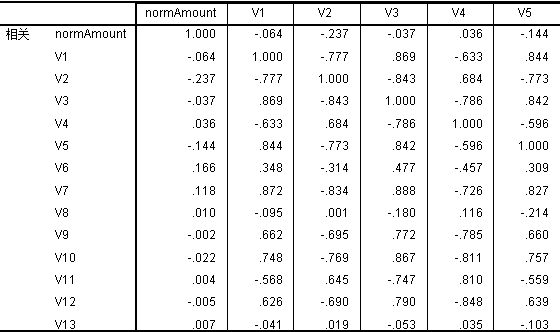
1. 主成分分析提取

基于字段 到 分析，由于28个主成分经过主成分分析提取得来，故28 个字段均无详细说明，在建模、验算过程中将其当作具有实际意义的指标进行运算。

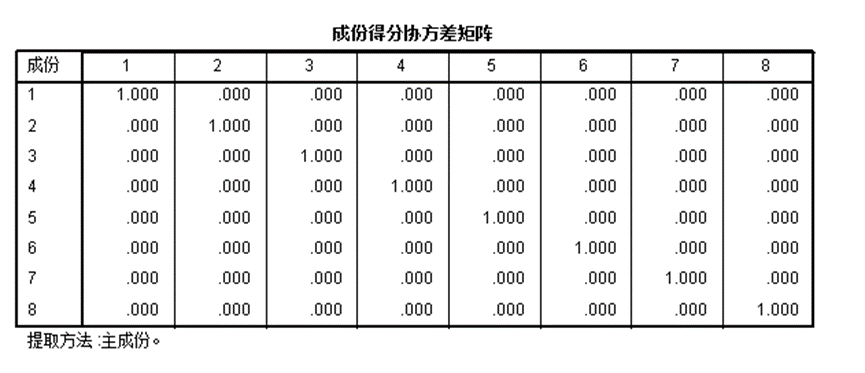
通过对数据集进行主成分分析，其步骤[5]如下：

* + 对数据集进行归一化处理（以上数据已处理完毕）
  + 计算归一化后的数据集的协方差矩阵
  + 计算协方差矩阵的特征值和特征向量
  + 保留最重要的个特征（通常要小于），也可以自己制定，也可以选择一个阈值，然后通过前个特征值之和减去后面个特征值之和大于这个阈值，则选择k
  + 找出个特征值对应的特征向量
  + 将的数据集乘以个维的特征向量的特征向量,得到最后降维的数据。

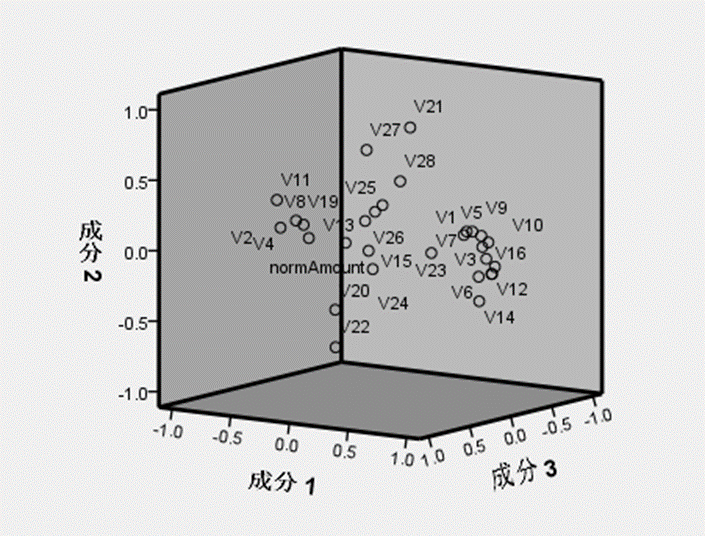
这里我们以数据集为例，进行了主成分分析，结果计算统计分析如下:



相关系数矩阵（部分）

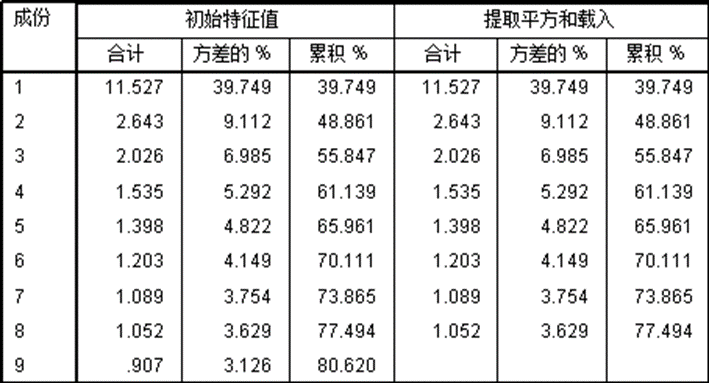


相关系数矩阵值

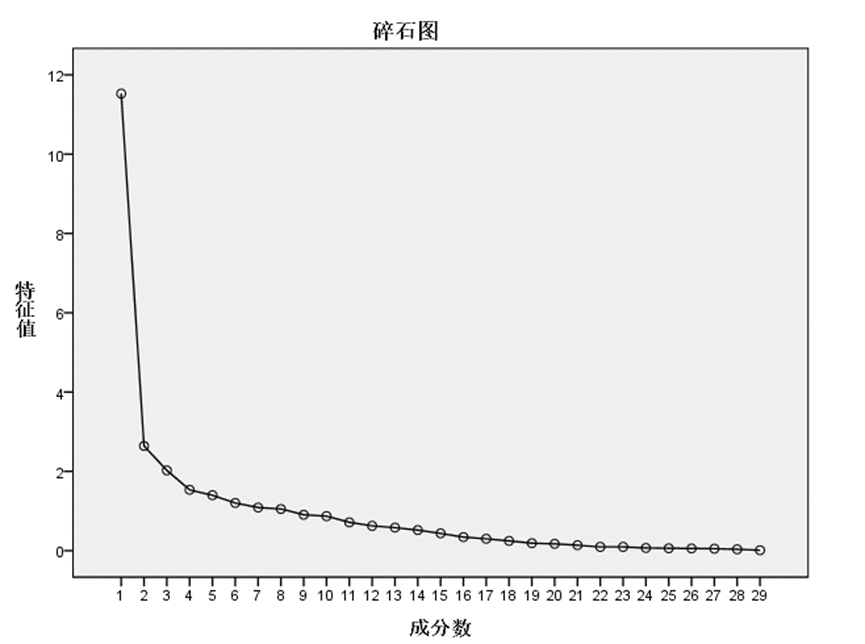


成分图

即p值，代表假设检验中的显著性，通常如果<0.05，拒绝虚无假设（原假设），接受备择假设，反之则无充分理由拒绝虚无假设，由图6，图7可见，对于相关分析，通常<0.05，这意味着相关系数有统计学意义，变量间的确存在相关。由图8可得出主成分之间具有一定的相关性，其中具有较强的相关性，说明主成分之间确实存在信息的冗余与重叠。



方差分解主成分提取分析表（部分）



方特征值碎石图

从图9、图10可得出的特征值大于1，因此我们选取这八个特征模型作为主成分。

初始因子载荷矩阵中的数据除以主成分相对应的特征值开平方根便得到八个主成分中每个指标所对应的系数（全表见附录五），可以得出主成分表达式,主成分表达式如附录六所示。

以每个主成分所对应的特征值占所提取主成分总的特征值之和的比例作为权重计算，可得到8个主成分模型，再与原先标准化的通过公式进行数据处理，得到全新的八个指标成为数据集详见附录七，通过主成分分析，对比原有数据损失比例为，但通过这八个指标来代替原先的29个指标，大大减少数据量的处理。

同理，我们可以得到数据集详见附录八，其数据数据损失比例为。

# 模型的建立与求解



## 模型一：基于单层决策树的集成学习模型的建立

通过前期的数据处理，我们采用决策树分类器中的单层决策树对欺诈交易数据与正常交易数据进行简单分类。单层决策树( )是一种简单的决策树，它仅仅是基于单个特征来做决策，由于这棵树只有一次分裂过程，因此它实际上仅仅是一个树桩。其算法步骤如下：

1. 按照横坐标或者纵坐标来划分，这里将横坐标视为特征一，纵坐标视为特征二（单决策树只能根据一个特征来做决策）；
2. 基于第一步，因此外层循环即为数据集特征的循环,按照第一个指标来划分类别，那么第一个特征点中存在节点，即树的左右分支，判断步骤如下：
3. 根据数据大小跟定一个阈值T，这里我们,即最小特征值（第一个坐标的最小值）+(1,2,3,4,5.....)\*步长，这个阈值随着整数值INT的改变而循环改变；大于阈值T的则为“右节点”，反之则为“左节点”;
4. 判断错误率，初始化一个5\*1的列向量e，全部为1，如果预测结果和标签相同，则将初始化对应的值修改为0，最后再用一个权重向量D.T\*e,这个值即为最后的错误率，如果这个错误率小于一定的阈值，即为最有效决策树。
5. 比较每个阈值T的结果是"右节点"和"左节点"的错误率，同时判断大循环中第一个特征的错误率，或者第二个特征的错误率；

通过计算得出简单分类器的正确识别率过低，为提高正确率，我们采用基于算法对其进行优化。

是一种迭代算法，其核心思想是针对同一个训练集训练不同的分类器(弱分类器)，然后把这些弱分类器集合起来，构成一个更强的最终分类器(强分类器)。基于单层决策树算法其运行过程如下：训练数据中的每个样本，并赋予其一个权重，这些权重构成了权重向量。一开始，这些权重都初始化成相等值。首先在训练数据上训练出一个弱分类器并计算该分类器的错误率，然后在同一数据集上再次训练弱分类器。在分类器的第二次训练当中，将会重新调整每个样本的权重，其中第一次分对的样本的权重将会降低，而第一次分错的样本的权重将会提高。为了从所有弱分类器中得到最终的分类结果，为每个分类器都分配了一个权重值，这些alpha值是基于每个弱分类器的错误率进行计算的。计算出alpha值之后，可以对权重向量D进行更新，以使得那些正确分类的样本的权重降低而错分样本的权重升高。在计算出权重向量之后，又开始进入下一轮迭代。算法会不断的重复训练和调整权重的过程，直到训练错误率为0或者弱分类器的数目达到用户的指定值为止[6]。

我们将数据处理所得到的指标作为训练集，即每个字段的数值都是一个训练样本，字段作为测试集，步骤如下所示：

1. 给定训练数据集：，其中用于表示训练样本的类别标签，。的目的就是从训练数据中学习一系列弱分类器或基本分类器，然后将这些弱分类器组成一个强分类器。
2. 首先初始化训练数据的权值分布。每个训练样本最开始都被赋予相同的权值:,这样训练样本集的初始权值分布：

(4)

1. 进行迭代
2. 选取一个当前误差率最低的弱分类器h作为第t个基本分类器Ht。并计算弱分类器，该弱分类器在分布D\_t上的误差为：

 (5)

由上述式子可知，在训练数据集上的误差率就是被误分类样本的权值之和。

1. 计算该弱分类器在最终分类器中所占的权值（弱分类器权重用表示）则

αt=ln（）

1. 更新训练样本的权值分布：

(6)

其中为归一化

1. 最后，按弱分类器权重αt组合各个弱分类器，即

f（x）= （7）

通过符号函数的作用得到一个强分类器为

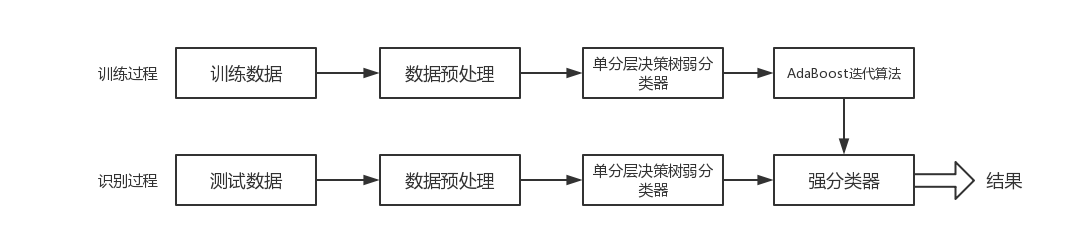
=sign(f(x))=sign()

（8）

## 模型一求解

本文对于消费记录进行是否存在消费欺诈的判定流程主要分为以下4步：

1. 数据预处理，将原先所有字段内容进行预处理，为下一步做准备，这里我们；②生成弱分类器，将数据预处理得到的八个指标通过单层决策树弱分类器方法生成八个弱分类器；③生成强分类器，通过迭代算法生成一个强分类器；
2. 识别：对需要识别的消费记录先进行预处理后用训练出来的强分类器进行判断是否存在欺诈消费行为，该模型对判定是否存在消费欺诈行为流程如图10



算法流程图



### **实验数据预处理**

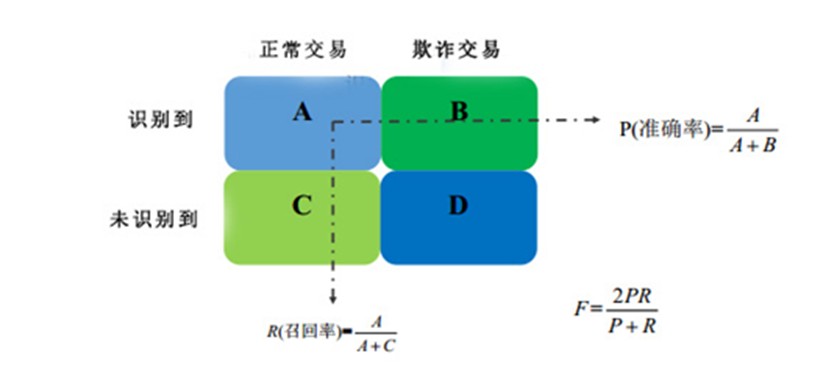
这里我们使用5.1.2最终得到的过采样数据集与下采样数据集。其中，两个数据集中的0，1比例均为1。并且随机选取数据集中的70%作为训练集，30%作为测试集。

### **定义指标对模型的分类能力进行表达**

模型建立过程中我们需要对模型的训练结果不断进行调整使其逐渐趋近于真实情况,因此需要些指标来对模型执行能力进行衡量,而数据挖掘分类领域两个最基本指标是召回率()和准确率( ),召回率也叫查全率,准确率也叫查准率,概念公式：

召回率()模型识别的正确的交易信息个数/模型识别的交易信息个数总数

准确率()模型识别的正确的交易信息个数/模型识别出的交易信息个数总数



准确率与召回率的关系

值得注意的是准确率和召回率是互相影响的,理想情况下肯定是做到两者都高,但是一般情況下准确率高、召回率就低,反之召回率低、准确率高。为了权衡二者的影响,我们引入了来综合模型。是和加权调和平均：



当参数α=1时，就是常见的F1，也即：



同时，在这里判断交易信息为欺诈，实际上为非欺诈交易会使银行失去该交易订单的获利；判断交易信息为正常，实际上为欺诈交易会使银行照成损失，因此，在对分类效果评判的标准中，误判率M也是一大重要指标，而误判率M可以表示如下：



其中FP为预测为1实际为0的计数，FN为预测为0实际为1的计数，是总计样本数。

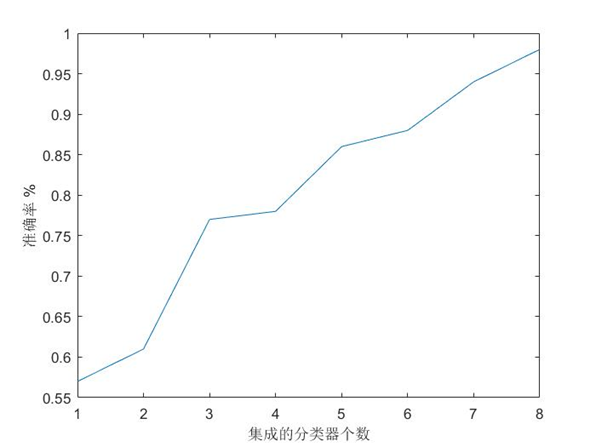
### **对弱分类器进行迭代**

采用建立的迭代算法模型，通过加权集成弱分类器而得出的强分类器，对附件数据集中的交易信息进行分类，最终通过算法模型计算出来的结果分析如下：

1. **过采样结果测试分析：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 混淆矩阵 | | 预测 | |  |
| 1 | 0 | 合计 |
| 实际 | 1 | 81804 | 3500 | 85304 |
| 0 | 1025 | 84252 | 85277 |
| 合计 | | 82829 | 87752 | 170581 |

1. 过采样结果测试分析



过采样集成分类器个数与准确率的关系

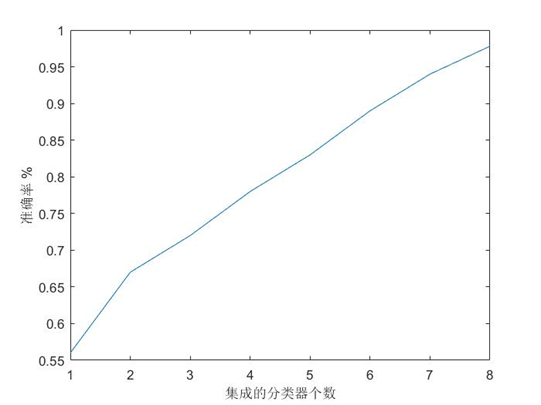
在过采样检验过程中，算法预测其存在欺诈交易且实际也存在欺诈交易的数量是81804条，预测其为正常交易而实际为欺诈交易的数量是3500条，预测其为欺诈交易而实际为正常交易的数量是1025条，预测其实正常交易且实际也为正常交易的数量是84252条，结果如上表1所示。

从图 12 得出，当我们集成的分类器个数（即弱分类器的个数）为一个时，我们预判的准确率在55%--60%区间，当我们增加集成分类器个数时，准确率虽变化速率不一，但总体呈现上升趋势，且最终趋近于96%。

1. **下采样结果测试分析**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 混淆矩阵 | | 预测 | |  |
| 1 | 0 | 合计 |
| 实际 | 1 | 132 | 9 | 141 |
| 0 | 3 | 146 | 149 |
| 合计 | | 135 | 155 | 290 |

1. 下采样结果测试分析



下采样集成分类器个数与准确率的关系

在下采样检验过程中，算法预测其存在欺诈交易且实际也存在欺诈交易的数量是132条，预测其为正常交易而实际为欺诈交易的数量是9条，预测其为欺诈交易而实际为正常交易的数量是3条，预测其实正常交易且实际也为正常交易的数量是146条，结果如表2所示。

从图13得出，当我们集成的分类器个数（即弱分类器的个数）为一个时，我们预判的准确率在 区间，当我们增加集成分类器个数时，准确率相对稳定的增长趋势增加，且最终趋近于。

**（3）综合两个数据集分析**

通过过采样、下采样两种数据检验方法，我们根据定义求出各项评价指标具体值，从下表可得出我们通过算法模型计算出来结果误判率均低于5%，且召回率趋近于，精准率趋近于，值均超过0.95，具体值表3所示：

|  |  |  |
| --- | --- | --- |
| 评价指标 | 过采样 | 下采样 |
| 召回率 | 0.958970 | 0.936170 |
| 精准率 | 0.987625 | 0.977778 |
|  | 0.973087 | 0.956522 |
| 误判率 | 2.65% | 4.13% |

1. 过采样、下采样各项评价指标具体值

通过以上分析，显然的算法模型计算出来结果比原先简单分类器所识别判断结果准确率显著提高，因此我们的模型具有强欺诈识别能力，且有较高的参考价值以及意义。

## 模型二：基于的的深度学习模型

自动编码器（）是一种多层前传神经网络，是一种无监督的学习算法，主要用于数据的降维或者特征的抽取。在深度学习中，自动编码器可用于在训练阶段开始前，确定权重矩阵的初始值。

在基于无监督学习算法时，通常是接入一个输入向量x进行编码，选取“”作为激活函数计算方式如式（9），再进行解码，得到一个编码结果y，该结果在解码器作用下得到重构向量z，计算公式如式（10）。该过程可以看作是对输入数据的压缩编码，将高维的原始数据用低维的向量表示，使压缩后的低维向量能保留输入数据的典型特征，从而能够较为方便的恢复原始数据[7]。





其中注意与一般的多层前传神经网络不同的是，前后两个公式使用的权重矩阵、互为转置。

区别于有监督学习的是，自动编码器的学习是无监督的，其目标是使得输出重构向量尽可能还原输入向量的状态，将重构带来的损失降至最低，本文所采取的损失函数如式(11):



其中矩阵表示是一个由个样本的向量组成的矩阵，表示 与 之间的散度，由于的维度比的维度小得多，因此可以学习到输入样本的低维表示，并且能通过解码尽可能涵盖与高维表示相同的信息。自动编码器权重的训练采用随机梯度下降算法，保证输出结果能够反映出输入层的主要特征，使用以下公式更新权重矩阵如式（12），　其中 η 为更新的步长，参数和采用相同方式更新



非线性降维方法途径是保持原始高维数据的某种局部结构或信息不变，直接找到一个低维矩阵y来代替高维数据矩阵x。而自动编码器强大的非线性表达能力使得它容易出现对输入数据过拟合的现象，倘若直接运用自动编码器算法, 将导致最后抽取的特征向量不能反映短文本的特点, 训练出来的模型泛化能力比较差, 无法推广运用到其他短文本中。因此, 我们对自动编器的学习能力进行约束，通过过采用变量选择法，来对模型进行约束、修正。对数据进行稀疏编码，从而保证算法中各项编码的稀疏性，具体将式（12）调整成式（13）式（14）进行计算[8]。





## 模型二求解



### **数据预处理**

由于自带降维处理，因此我们不需要使用处理后的数据集，这里我们使用5.1.2为PCA处理的数据集,与数据集。同样，两个数据集中的0，1比例均为1，并且我们使用随机选取数据集中的 作为训练集， 作为测试集。

### **搭建模型**

这里我们基于库与的开源顶层框架，搭建模型，我们设置29个神经元作为输入层，分别建立4个完全连接层，分别为28，10，10，29个神经元。前两层用于编码器，最后两层用于解码器。训练期间将使用L1正规化。同时编码器和解码器层分别采用了“”作为非线性激活函数。训练模型设定为150个，批量大小为32个样本，并将最佳性能模型点保存作为训练好的文件输出。

### 加载训练模型对测试数据进行识别

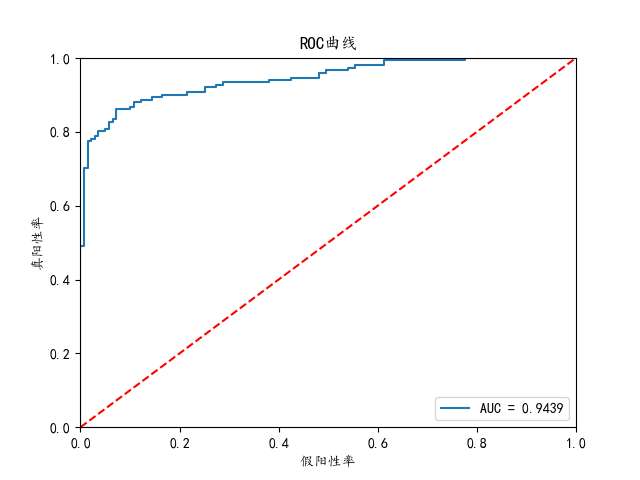
根据训练好的模型，我们统计了下采样、过采样中的预测数据和测试数据以及分类正确的数目的各项指标的具体值，结果如表4所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | 下采样数据集 | | 过采样数据集 | |
| 数据类别 | 预测数据 | 测试数据 | 测试数据 | 测试数据 |
|  | 290 | 290 | 170581 | 170581 |
|  | 14.23425 | 0.52069 | 9.738905 | 0.498244 |
|  | 31.669156 | 0.500435 | 19.088655 | 0.499990 |
|  | 0.166332 | 0 | 0.073276 | 0 |
| 25% | 0.499845 | 0 | 0.398729 | 0 |
| 50% | 1.04733 | 1 | 1.108408 | 0 |
| 75% | 9.2717 | 1 | 9.115259 | 1 |
|  | 266.328541 | 1 | 301.323782 | 1 |

1. 下采样、过采样各项指标的具体值

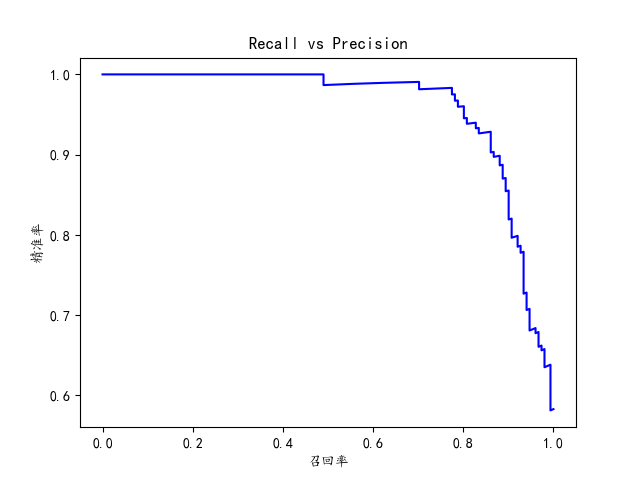
同时，针对模型一定义的指标，我们对识别结果进行分析。在分析之前，引入曲线概念。曲线是根据一系列不同的二分类方式（分界值或决定阈），以真阳性率（灵敏度）为纵坐标，假阳性率（1-特异度）为横坐标绘制的曲线。传统的判断试验评价方法有一个共同的特点，必须将试验结果分为两类，再进行统计分析。曲线的评价方法与传统的评价方法不同，无须此限制，而是根据实际情况，允许有中间状态，可以把试验结果划分为多个有序分类，因此，曲线评价方法对评估模型的分类能力与准确率更为适用。

1. **下采样测试数据**



下采样累计风险评估曲线图

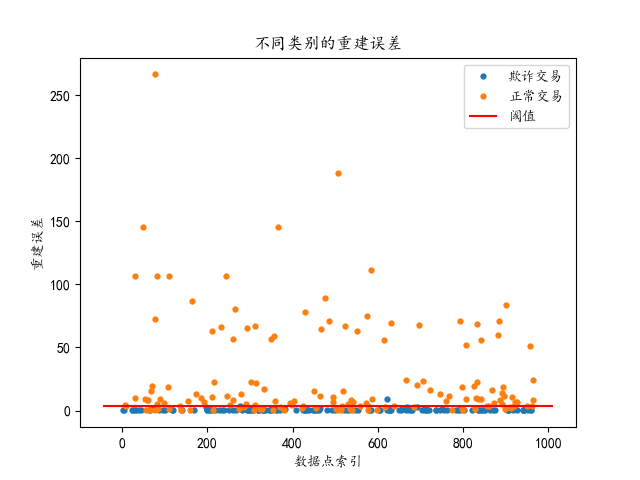
通过下采样检验，曲线越凸越近左上角表明其分类能力越大，同时曲线下面积可评价诊断准确性，根据上图14可知，该训练模型的模型评估曲线达预期效果，说明模型有效，可以较好的侦测到欺诈交易信息，准确度；



召回率与精准率的比较

此后我们在对精准率和召回率进行比较，从图15可直观得出召回率与精准率呈负相关关系，即召回率越高，精准率越低，反之召回率越低，精准率越高

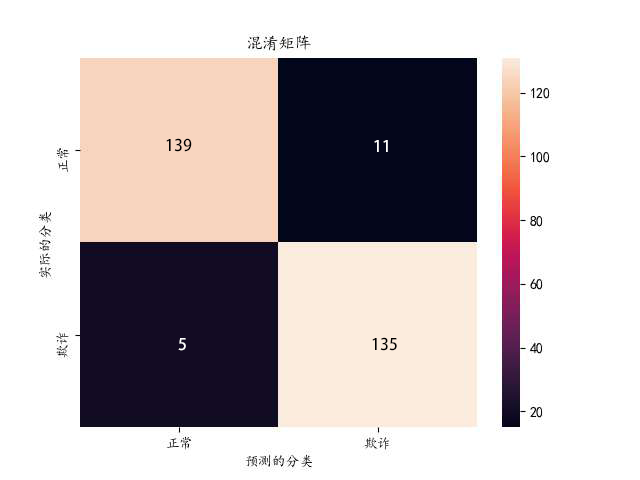
为了预测新/不可见的信用卡交易是否正常或欺诈，我们可以从交易数据本身计算重建错误。如果错误大于预定阈值，我们将其标记为欺诈，设定阀值：，预测情况如下：



下采样的不同类别的重建误差

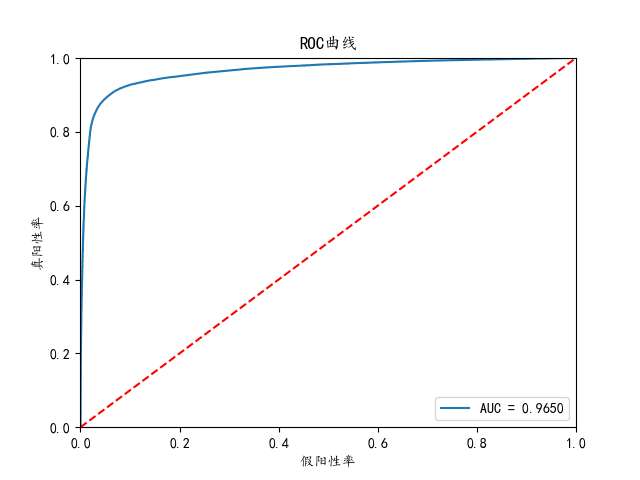
对不同类别的重建误差构建如下的混淆矩阵，其中预测其为欺诈实际也为欺诈共135个，预测其为正常实际也为正常的共139个，从错误分类矩阵可以得出

分类结果准确率达94.482%，是参考意义较大的优的分类鉴别器。



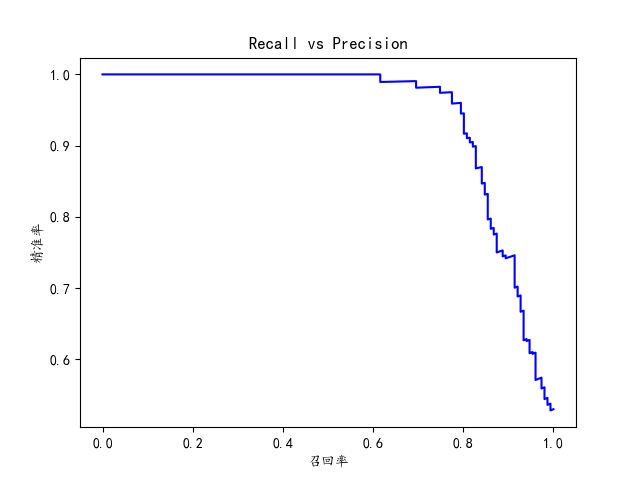
下采样对不同类别的重建误差构建如下的混淆矩阵

1. **过采样测试数据**



过采样累计风险评估曲线图

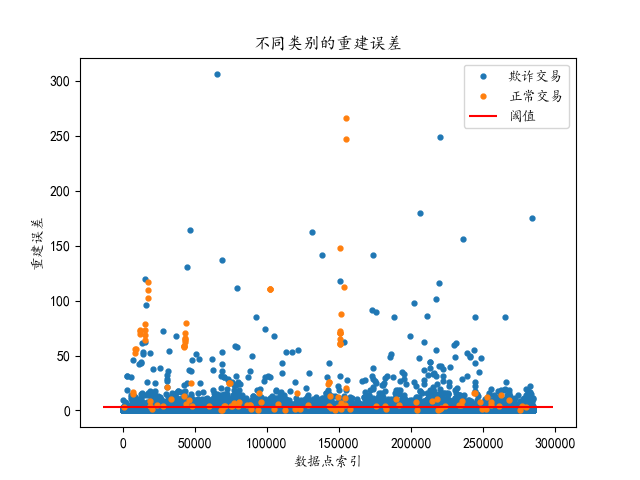
通过过采样检验，曲线越凸越近左上角表明其分类能力越大，同时曲线下面积可评价诊断准确性，根据上图18可知，该训练模型的模型评估曲线达预期效果，说明模型有效，可以较好的侦测到欺诈交易信息，准确度



召回率与精准率的比较

此后我们在再次对精准率和召回率进行比较，从图19可直观得出召回率与精准率呈负相关关系，即召回率越高，精准率越低，反之召回率越低，精准率越高

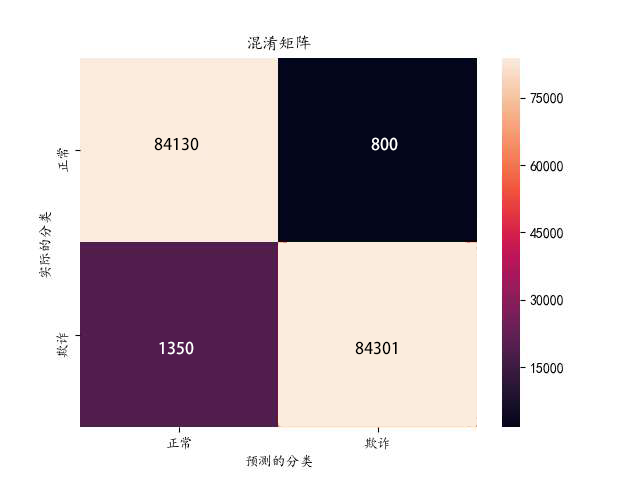
为了预测新/不可见的信用卡交易是否正常或欺诈，我们可以从交易数据本身计算重建错误。如果错误大于预定阈值，我们将其标记为欺诈，设定阀值：，预测情况如下：



过采样的不同类别的重建误差

对不同类别的重建误差构建如下的混淆矩阵，其中预测其为欺诈实际也为欺诈共84301个，预测其为正常实际也为正常的共84130个，从错误分类矩阵可以得出

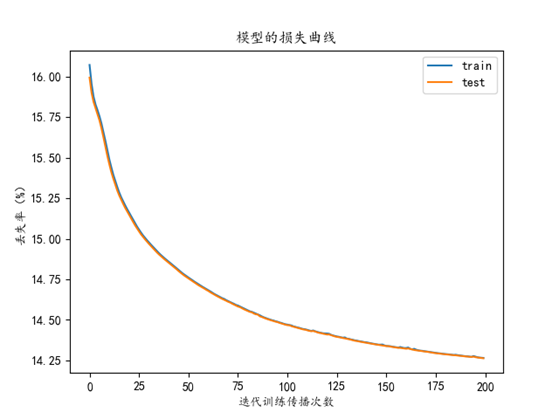
分类结果准确率达，是参考意义较大的优的分类鉴别器。



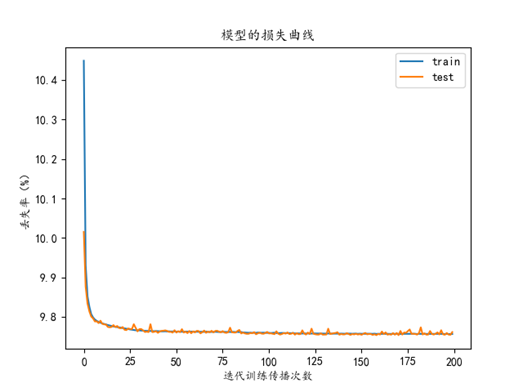
过采样对不同类别的重建误差构建如下的混淆矩阵

# 鲁棒性检验

在上文中,我们我们在模型二在确定值时假设为150，选择对于这些主观选择的因素,我们对其进行鲁棒性[9]分析来观察当值与我们假定的值不同时对模型结果的影响来观察模型是否鲁棒。为了观察效果,我们直接做出过采样和下采样假定值与结果的函数图像,如下所示。



模型的损失曲线-下采样



模型的损失-过采样

如图 22 ，图 23 下采样模型的损失稳定后基本上维持在0.145以下，可以看出经过200个迭代较好收敛，过采样模型的损失基本上维持在0.097以下，可以看出经过50~200个迭代趋于较好收敛。为了模型的通用，我们选取150作为值能达到较好的收敛效果

# 模型评价



## 模型一与模型二的比较

我们以过采样数据集为例，对模型一和模型二的基本原理，训练时间、F值、误判率等进行了对比结果如下表所示：

|  |  |  |
| --- | --- | --- |
| (以过采样数据集为例) | 模型一 | 模型二 |
| 学习方法 | 传统机器学习 | 深度学习 |
| 训练时间 | 276s | 3123s |
|  | 0.973087 | 0.9874 |
| 误判率 | 2.65% | 1.73% |
| 系统参数调整（阈值） | 自适应 | 人工调整 |
| 时间复杂度 |  |  |
| 空间复杂度 |  |  |
| 数据损失 | 22.5% | 0 |

1. 模型一与模型二对比

## 模型的优、缺点

|  |  |  |
| --- | --- | --- |
|  | 优点 | 缺点 |
| 模型一 | 针对训练集的八个指标制作成的八个弱分类器，我们使用迭代算法将其聚合成为一个强分类器，进而进行综合评定，最终计算得出我们的识别率，对比使用单个分类器的效果会有明显的提高。同时，算法通过加权集成弱分类器构成强分类器，是一个标准的框架算法，当我们对现有特征工程理解或者分析比较浅薄时，在以后的工作中，我们可以在原有的分类器的基础上重新增加新指标的分类器作为弱分类器，不需要改动其他部分的算法，特别适合小型企业更新学习，其扩展性能十分强大。 | 添加指标制作分类器，由于主成分的特性，所有被主成分的交易指标数据都要更新，这样会使全部改变，计算成本较大。 |
| 模型二 | 自带数据降维程序，可以极大利用数据集，避免原始数据的丢失；高效率，使用训练好的模型可以节省了大量的计算；可塑性强：对于不同的数据集只需要调整参数，就能改变模型。具有很强的灵活性和成长性。一个程序可以持续改进，然后达到完美的境界；普适性：模型二通过算法用来调整参数，做非常复杂的非线性映射，通过学习来解决问题，可以根据问题自动建立模型，所以能够适用于各种问题，而不是局限于某个固定的问题。适合大型企业分布式更新计算学习。 | 计算量大，训练耗时，模型正确性验证复杂且麻烦；黑箱算法，对于各指标的参数调整没法能直接解释。 |

1. 模型一与模型二优缺点对比

## 模型的改进和推广



### **模型一改进与推广**

采用单层决策树简单二值分类器作为弱分类器，在每个特征上进行训练时得到的阈值，不一定能很好的对正常评论和垃圾评论进行区分， 为此需要寻找识别效果更好的弱分类器， 例如利用分类器或者鉴别分析分类器，从而来提高识别欺诈交易数据的能力。通过构造弱分类器迭代成为强分类器，可以推广到处理，图像处理或推荐系统，例如通过处理得到各个特征指标，通过迭代自适应选取参数，进而做到精准识别。

### **模型二改进与推广**

使用自编码模型强大的非线性表达能力会使得模型经常出现对输入数据多度拟合的现象，我们在模型二中加入了范式正则化对模型进行改进，我们利用绝对值函数作为惩罚项来压缩自动编码器的系数，随着范式中参数 的值变大，即惩罚力度变大，训练得到的结果会变得稀疏，从而帮助模型达到拟合能力和泛化能力的均衡，这里我们为了帮助模型快速稀疏化，我们可以在输入层加入噪音，同时利用随机梯度下降算法避免模型受到个性化或者无关输入的影响，这样可以使我们的训练模型快速收敛，同时增强了模型的对拟合与泛化的均衡能力。

# 参考文献

[1]李睿. 信用卡犯罪研究[D].华东政法大学,2009.

[2]徐小军,王友仁,陈帅.基于下采样分数阶小波变换的图像融合新方法[J].仪器仪表学报,2014,35(09):2061-2069.

[3]张新刚,王泽忠.基于过采样技术提高数据采集精度的新方法[J].电力系统自动化,2004(14):58-62.

[4]孙红卫,吕春燕,祁爱琴,曹国华,韩春蕾.综合评价中数据标准化的原理研究[J].中国卫生统计,2015,32(02):342-344+349.

[5]林海明,杜子芳.主成分分析综合评价应该注意的问题[J].统计研究,2013,30(08):25-31.

[6]黄铃. 基于AdaBoost-LC的微博垃圾评论识别研究[D].重庆大学,2014.

[7]邓俊锋,张晓龙.基于自动编码器组合的深度学习优化方法[J].计算机应用,2016,36(03):697-702.

[8]刘勘,袁蕴英.基于自动编码器的短文本特征提取及聚类研究[J].北京大学学报(自然科学版),2015,51(02):282-288.

[9]吕大刚,宋鹏彦,崔双双,王闽雄.结构鲁棒性及其评价指标[J].建筑结构学报,2011,32(11):44-54.

# 附录

**SMOTE过采样算法(python)**

import random

from sklearn.neighbors import NearestNeighbors

import numpy as np

class Smote:

"""

SMOTE过采样算法.

Parameters:

-----------

k: int

选取的近邻数目.

sampling\_rate: int

采样倍数, attention sampling\_rate < k.

newindex: int

生成的新样本(合成样本)的索引号.

"""

def \_\_init\_\_(self, sampling\_rate=5, k=5):

self.sampling\_rate = sampling\_rate

self.k = k

self.newindex = 0

def fit(self, X, y=None):

if y is not None:

negative\_X = X[y == 0]

X = X[y == 1]

n\_samples, n\_features = X.shape

# 初始化一个矩阵, 用来存储合成样本

self.synthetic = np.zeros((n\_samples \* self.sampling\_rate, n\_features))

# 找出正样本集(数据集X)中的每一个样本在数据集X中的k个近邻

knn = NearestNeighbors(n\_neighbors=self.k).fit(X)

for i in range(len(X)):

k\_neighbors = knn.kneighbors(X[i].reshape(1, -1),return\_distance=False)[0]

# 对正样本集(minority class samples)中每个样本, 分别根据其k个近邻生成

# sampling\_rate个新的样本

self.synthetic\_samples(X, i, k\_neighbors)

if y is not None:

return (np.concatenate((self.synthetic, X, negative\_X), axis=0),

np.concatenate(([1] \* (len(self.synthetic) + len(X)), y[y == 0]), axis=0))

return np.concatenate((self.synthetic, X), axis=0)

# 对正样本集(minority class samples)中每个样本, 分别根据其k个近邻生成sampling\_rate个新的样本

def synthetic\_samples(self, X, i, k\_neighbors):

for j in range(self.sampling\_rate):

# 从k个近邻里面随机选择一个近邻

neighbor = np.random.choice(k\_neighbors)

# 计算样本X[i]与刚刚选择的近邻的差

diff = X[neighbor] - X[i]

# 生成新的数据

self.synthetic[self.newindex] = X[i] + random.random() \* diff

self.newindex += 1

# ------通过过采样获取284302条calss为1的数据

import pandas as pd

import csv

df=pd.read\_csv('creditcard\_data.csv')

data = []

for i, element in enumerate(df['Class']):

if element == 1:

data.append(df.iloc[i, :])

X = np.array(data)

smote = Smote(sampling\_rate=588, k=483)

data1=smote.fit(X).tolist()

#保存为csv

csvFile2 = open('data\_Over sampling.csv','w', newline='') # 设置newline，否则两行之间会空一行

writer = csv.writer(csvFile2)

m = len(data1)

for i in range(m):

writer.writerow(data1[i])

csvFile2.close()

arr1=smote.fit(X)

arr1.to\_csv('data\_Over sampling.csv')

# ---------------------------

# 提取class为1的数

import csv

import random

data=[]

for i, element in enumerate(df['Class']):

if element==1:

data.append(df.iloc[i,:])

csvFile2 = open('csvFile2.csv','w', newline='') # 设置newline，否则两行之间会空一行

writer = csv.writer(csvFile2)

m = len(data)

for i in range(m):

writer.writerow(data[i])

csvFile2.close()

import csv

import random

import pandas as pd

df=pd.read\_csv('creditcard\_data.csv')

data=[]

count=0

for i, element in enumerate(df['Class']):

if element==0:

data.append(df.iloc[i,:])

count+=1

if count==284487:

break

csvFile2 = open('over.csv','a', newline='') # 设置newline，否则两行之间会空一行

writer = csv.writer(csvFile2)

m = len(data)

for i in range(m):

writer.writerow(data[i])

csvFile2.close()

**数据标准化(python)**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('creditcard\_data.csv')

# data.head(10)

#

print(data.shape)

count\_class = pd.value\_counts(data['Class'],sort = True).sort\_index()

print (count\_class)

from sklearn.preprocessing import StandardScaler #导入数据预处理模块

data['normAmount'] = StandardScaler().fit\_transform(data['Amount'].values.reshape(-1,1)) # -1表示系统自动计算得到的行，1表示1列

data['normTime'] = StandardScaler().fit\_transform(data['Time'].values.reshape(-1,1))

# data = data.drop(['Time','Amount'],axis = 1) # 删除两列，axis =1表示按照列删除，即删除特征。而axis=0是按行删除，是删除样本

#print (data.head(3))

# print (data['normAmount'])

# print(type(data['normTime']))

# #字典中的key值即为csv中列名

# dataframe = pd.DataFrame({'normTime':data['normTime'],'normAmount':data['normAmount']})

#

# #将DataFrame存储为csv,index表示是否显示行名，default=True

# dataframe.to\_csv("test.csv",index=False,sep=',')

**主成分算法(matlab)**

function [] = pca(data,allData,n)

[y,x] = size(data);

[y\_alldata,x\_alldata] = size(allData);

result = zeros(y\_alldata,8);

for i=1:x

for i2=1:y

fprintf('正在处理第 %d 行 %d 列\n',i2,i);

t\_data = data(:,i)';

for i3=1:y\_alldata

t\_alldata = allData(i3,:);

result(i3,i) = sum(t\_data.\*t\_alldata) \* n;

end

end

end

csvwrite('111/下采样8个指标.csv',result)

end

**打乱数据集顺序代码(matlab)**

function [] = downsampling(data, resulrFilename)

result = [];

randomNUM = [];

len = length(data(:,1));

for i=1:483

while(1)

num = fix(rand(1) \* len) + 1;

fprintf('%d\n',num)

if any(randomNUM==num)==0 && data(num,31)==0

break

end

end

result = [result;data(num,:)];

csvwrite(resulrFilename,result)

end

**模型一adaboost代码(matlab)**

function [same] = AdaBoostTest()

fprintf('读取数据中...\n');

csv\_data = csvread('111/下采样8个指标乱序.csv', 1, 0);

fprintf('读取完成.\n');

same = zeros(8,1);

len = fix(length(csv\_data(:,1)));

train\_len = fix(len \* 0.7);

train\_data = csv\_data(1:train\_len,1:8);

train\_label = csv\_data(1:train\_len,9);

test\_data = csv\_data(train\_len+1:len,1:8);

test\_label = csv\_data(train\_len+1:len,9);

fprintf('下采样\n')

for i=1:8

ens = fitensemble(train\_data(:,1:i),train\_label,'AdaBoostM1' ,100,'tree','type','classification');

predict\_label = predict(ens, test\_data(:,1:i));

TP = 0;

FN = 0;

FP = 0;

TN = 0;

for i2=1:length(predict\_label)

if test\_label(i2,1) == 1 && predict\_label(i2,1) == 1

TP = TP + 1;

end

if test\_label(i2,1) == 1 && predict\_label(i2,1) == 0

FN = FN + 1;

end

if test\_label(i2,1) == 0 && predict\_label(i2,1) == 1

FP = FP + 1;

end

if test\_label(i2,1) == 0 && predict\_label(i2,1) == 0

TN = TN + 1;

end

if test\_label(i2,1) == predict\_label(i2,1)

same(i,1) = same(i,1) + 1;

end

end

same(i,1) = same(i,1)/length(predict\_label)\*100;

fprintf('%d 列：\n',i);

fprintf('TP: %f FN: %f\nFP: %f TN: %f\n',TP,FN,FP,TN);

P = TP / (TP + FP);

R = TP / (TP + FN);

fprintf('P: %f R: %f\nF1: %f\n',P,R,2\*P\*R/(P+R));

fprintf('准确率: %f %%\n',same(i,1));

end

end

**模型二自动编码器算法(python)**

import pylab

import numpy as np

import pickle

import matplotlib.pyplot as plt

from scipy import stats

import tensorflow as tf

import seaborn as sns

from pylab import rcParams

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

from keras.layers import Input,Dense

from keras.models import Model,load\_model

from keras.callbacks import ModelCheckpoint,TensorBoard

from keras import regularizers

import pandas as pd

from pylab import mpl

import time

mpl.rcParams['font.sans-serif'] = ['KaiTi'] # 指定默认字体

mpl.rcParams['axes.unicode\_minus'] = False # 解决保存图像是负号'-'显示为方块的问题

data =pd.read\_csv('creditcard\_data.csv')

#查看数据格式

# print(df.head())

#查看数据结构

# print(df.shape)

count\_classes = pd.value\_counts(data['Class'],sort = True).sort\_index()

count\_classes.plot(kind='bar',rot=0)

plt.title('class的分布情况')

plt.xlabel('Class')

plt.ylabel('数目')

plt.show()

#

RANDOM\_SEED=42

x\_train,x\_test=train\_test\_split(data,test\_size=0.3,random\_state=RANDOM\_SEED)

x\_train=x\_train[x\_train.Class==0]

x\_train=x\_train.drop(['Class'],axis=1)

y\_test=x\_test['Class']

x\_test=x\_test.drop(['Class'],axis=1)

x\_train=x\_train.values

x\_train=x\_test.values

print(x\_train.shape)

input\_dim=x\_train.shape[1]

encoding\_dim=14

input\_layer=Input(shape=(input\_dim ,))

encoder=Dense(encoding\_dim,activation='tanh',activity\_regularizer=regularizers.l1(10e-5))(input\_layer)

encoder=Dense(int(encoding\_dim/2),activation='relu')(encoder)

decoder=Dense(int(encoding\_dim/2),activation='tanh')(encoder)

decoder=Dense(input\_dim,activation='relu')(decoder)

autoencoder=Model(inputs=input\_layer,outputs=decoder)

nb\_epoch=100

batch\_size=32

autoencoder.compile(optimizer='adam',loss='mean\_squared\_error',metrics=['accuracy'])

checkpointer=ModelCheckpoint(filepath='creditcard\_data.h5',verbose=0,save\_best\_only=True)

tensorboard=TensorBoard(log\_dir='.logs',histogram\_freq=0,write\_graph=True,write\_images=True)

history=autoencoder.fit(x\_train,x\_train,epochs=nb\_epoch,batch\_size=batch\_size,shuffle=True,validation\_data=(x\_test,x\_test),verbose=1,callbacks=[checkpointer,tensorboard]).history

print(history)

#

plt.plot(history['loss'])

plt.plot(history['val\_loss'])

plt.title('模型的损失曲线')

plt.ylabel('丢失率（%）')

plt.xlabel('迭代训练传播次数')

plt.legend(['train','test'],loc='upper right')

plt.show()

# data\_Over sampling 过采样模型

# data\_Lower sampling 下采样模型

autoencoder=load\_model('data\_Over sampling.h5')

prediction=autoencoder.predict(x\_test)

mse=np.mean(np.power(x\_test-prediction,2),axis=1)

error\_df=pd.DataFrame({'reconstruction\_error':mse,'true\_class':y\_test})

print(error\_df.describe())

from sklearn.metrics import (confusion\_matrix,precision\_recall\_curve,auc,roc\_curve,recall\_score,classification\_report,precision\_recall\_fscore\_support)

fpr,tpr,thresholds=roc\_curve(error\_df.true\_class,error\_df.reconstruction\_error)

# ROC 曲线

roc\_auc = auc(fpr, tpr)

plt.title('ROC曲线')

plt.plot(fpr, tpr, label='AUC = %0.4f'% roc\_auc)

plt.legend(loc='lower right')

plt.plot([0,1],[0,1],'r--')

plt.xlim([-0.001, 1])

plt.ylim([0, 1.001])

plt.ylabel('真阳性率')

plt.xlabel('假阳性率')

plt.show()

#精确率和召回率

precision, recall, th = precision\_recall\_curve(error\_df.true\_class, error\_df.reconstruction\_error)

plt.plot(recall, precision, 'b', label='Precision-Recall curve')

plt.title('Recall vs Precision')

plt.xlabel('召回率')

plt.ylabel('精准率')

plt.show()

#准确率

plt.plot(th, precision[1:], 'b', label='阈值-精准率曲线')

plt.title('不同阈值的精准率')

plt.xlabel('阈值')

plt.ylabel('精准率')

plt.show()

#召回率f

plt.plot(th, recall[1:], 'b', label='阈值-召回率曲线')

plt.title('不同阈值的召回率')

plt.xlabel('阈值')

plt.ylabel('召回率')

plt.show()

threshold=2.9

y\_pred = [1 if e > threshold else 0 for e in error\_df.reconstruction\_error.values]

conf\_matrix = confusion\_matrix(error\_df.true\_class, y\_pred)

groups = error\_df.groupby('true\_class')

fig, ax = plt.subplots()

for name, group in groups:

ax.plot(group.index, group.reconstruction\_error, marker='o', ms=3.5, linestyle='', label= '正常交易' if name == 1 else '欺诈交易')

ax.hlines(threshold, ax.get\_xlim()[0], ax.get\_xlim()[1], colors='r', zorder=100, label='阈值')

ax.legend()

plt.title('不同类别的重建误差')

plt.ylabel('重建误差')

plt.xlabel('数据点索引')

plt.show()

LABELS=['正常','欺诈']

#

plt.figure()

sns.heatmap(conf\_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt='d')

plt.title('混淆矩阵')

plt.ylabel('实际的分类')

plt.xlabel('预测的分类')

plt.show()

# #

TP=0

FN=0

FP=0

TN=0

f1data=[]

thresho=[]

for i in range(0,500):

threshold = i\*0.01

thresho.append(threshold)

y\_pred = [1 if e > threshold else 0 for e in error\_df.reconstruction\_error.values]

conf\_matrix = confusion\_matrix(error\_df.true\_class, y\_pred)

for T, P in zip(y\_pred, y\_test):

if T == 1 and P == 1:

TN += 1

if T == 1 and P == 0:

FN += 1

if T == 0 and P == 1:

FP += 1

if T == 0 and P == 0:

TP += 1

#

# print('TP' + str(TP))

# print('FN' + str(FN))

# print('FP' + str(FP))

# print('TN' + str(TN))

p = (TP + TN) / (TP + FN + FP + TN)

print(str(p))

f1data.append(p)

plt.xlabel("阈值") # X轴的文字

plt.ylabel("F1") # Y轴的文字

plt.title("阈值-F1曲线") # 图表的标题

plt.plot(thresho, f1data)

plt.show() # 显示图片

# f1data=[]

# thresho=[]

# for i in range(0,500):

# threshold = i\*0.01

# thresho.append(threshold)

# y\_pred = [1 if e > threshold else 0 for e in error\_df.reconstruction\_error.values]

# conf\_matrix = confusion\_matrix(error\_df.true\_class, y\_pred)

# # plt.figure(figsize=(12, 12))

# # sns.heatmap(conf\_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt='d')

# # plt.title('Confusion matrix')

# # plt.ylabel('True class')

# # plt.xlabel('Predicted class')

# # plt.show()

# for T, P in zip(y\_pred, y\_test):

# if T == 1 and P == 1:

# TN += 1

# if T == 1 and P == 0:

# FN += 1

# if T == 0 and P == 1:

# FP += 1

# if T == 0 and P == 0:

# TP += 1

#

# # print('TP' + str(TP))

# # print('FN' + str(FN))

# # print('FP' + str(FP))

# # print('TN' + str(TN))

# try:

# P = TN / (TN + FP)

# R=TN/(TN+FN)#所有准确的条目有多少被检索出来

# # R = TN / (TN + FN) # 所有欺诈的条目有多少被检索出来

# F = (P \* R) / (P + R)

# F1 = (2 \* P \* R) / (P + R)

# print('--------' + 'i:' + str(threshold))

# print("Precision:" + str(P)) # 检索出来的条目中（比如网页）有多少是准确的

# print("Recall:" + str(R))

# print('F1:' + str(F1))

# f1data.append(F1)

# except:

# pass

#

# plt.xlabel("阈值") # X轴的文字

# plt.ylabel("F1") # Y轴的文字

# plt.title("阈值-F1曲线") # 图表的标题

#

#

# plt.plot(thresho, f1data)

# plt.show() # 显示图片