电子商务导论

第6章 社会化商务用户欺诈检测

朱桂祥 (9120201070@nufe.edu.cn)
南京财经大学信息工程学院
江苏省电子商务重点实验室
电子商务信息处理国家级国际联合研究中心
电子商务交易技术国家地方联合工程实验室

南京大学校训

自建
“
入
会
计
中
中

目 录
Contents

第一节

背景及简介

第二节

恶意用户检测方法

第三节

案例

一、恶意用户检测简介

1. 简介

社会商务恶意用户是指那些由商业利益驱动，为达到如影响正常用户购买行为、扰乱商务环境等不正当目的，通过操纵软件机器人或水军账号，在电子商务网站中制造、传播虚假评论和垃圾信息等恶意用户的总称。

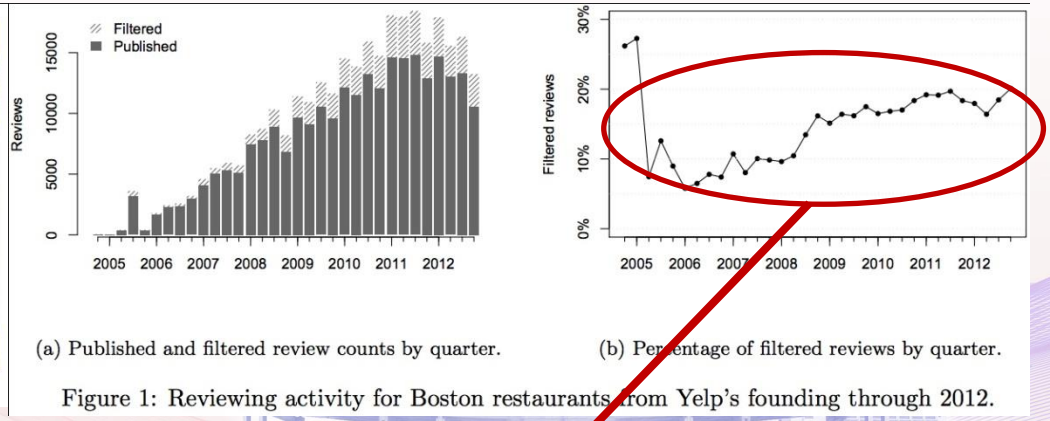
2002年，Amazon公司受到攻击，发现一起恶意用户利用恶意攻击使得网站在推荐一本基督教名著时还会推荐一本性方面的书籍

一、恶意用户检测简介

2. 恶意用户检测背景

受商业利益驱动，电子商务恶意用户通过发布和传播虚假商评达到影响正常用户购买行为（包括：推动自己商品的销售、打压竞争对手的信誉等），扰乱电子商务环境。

目前国外**Yelp、Amazon、eBay、hotel.com**以及国内的淘宝、京东、大众点评等电子商务网站均不同程度的受到恶意用户的攻击。



(a) Published and filtered review counts by quarter. (b) Percentage of filtered reviews by quarter.

Figure 1: Reviewing activity for Boston restaurants from Yelp's founding through 2012.

据哈佛商学院报道2013年Yelp的虚假评论达20%

Yelp评论至2007年以来被过滤掉评论比例逐年上升

一、恶意用户检测简介

3. 电子商务欺诈

定义：电子商务欺诈是以粉饰，虚构或是扭曲商品信息的途径，提高业绩，诱导消费者购买，提高商品排名等变相获取流量。

刷单产业：
有很多的网络店家雇人刷单，而所付出的成本，也的确是每单5-10元。但是这个刷单行业，也已经形成了产业化、规模化。

刷单属于伪造信用，是为了让新来的顾客通过浏览交易记录和买家评论，判定这家店铺商品的好坏，大量的购买记录和铺天盖地的好评会让不熟悉这家店铺的新顾客第一时间认定这家卖的是好货，从而产生真实的购买。

一、恶意用户检测简介

3. 电子商务欺诈

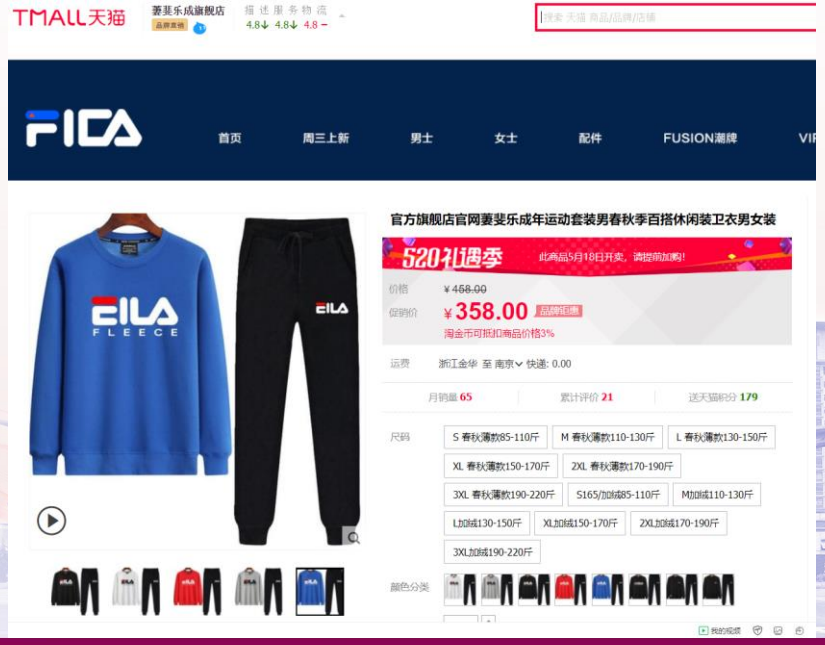
电子商务欺诈的危害：

(1) 对社会财产造成危害，使得资源不能有效配置。

(2) 危害了电子商务的信用，使人们对平台的信心受到打击。

(3) 隐蔽性多次进行，其实际结果比普通经济犯罪的危害更大。

淘宝的假货问题和京东的二手商品问题，存在部分的电商欺诈问题。



一、恶意用户检测简介

4. 电子商务反欺诈

目标：

(1) 对电子商务欺诈的一种识别服务。

(2) 使用评论数据，用户数据对电子商务中的恶意差评，刷单行为进行识别。

具体内容：

(1)电子商务推荐系统的恶意用户检测。

(2)电子商务网站恶意评论用户检测。

(3)社会化商务恶意用户检测。

一、恶意用户检测简介

4. 电子商务反欺诈

(1)托攻击：伪造用户，使得该虚假用户成为很多正常用户的近邻。

(2) 使得推荐系统频繁推荐自己的商品，而减少或是不推荐竞争对手的商品。

(3)托攻击的分类：

推攻击(恶意提高排名)

核攻击(恶意降低排名)

扰乱攻击(恶意歪曲排名)

(4)托攻击的检测算法

托攻击的本质是一个分类问题

分类问题和回归问题的本质区别是？

分类问题：分类问题的输出是离散型变量(如: 0，1，2)，是一种定性输出。(预测明天天气是阴、晴还是雨)

回归问题：回归问题的输出是连续型变量，是一种定量输出。(预测明天的温度是多少度)。

一、恶意用户检测简介

5.推荐系统的恶意用户检测

试图从推荐系统角度研究恶意用户行为和检测方法

一、恶意用户检测简介

5.推荐系统的恶意用户检测

-

The Process of User-CF

Scan all users, find similar interest users, use their scores to predict the scores of user u.

1. Establish a $m \times n$ matrix
2. Search neighbors using similarity algorithm
3. Recommendation generation

推

-

号，

商品评分	1	2	3	4	5	6	7	8	与Alice的相似度
Alice	5	3	3	4	2	1			\
User 1	3	4	2	3	4	5	1	3	-0.67
Use 2	4	3	1	2	4	2	4	1	0.23
User 3	4	2	1	3	4	1	5	5	0.52

Alice和User 3的相似度最高，而User 3对商品7、8打最高分，所建将商品7、8推荐给Alice.

一、恶意用户检测简介

5.推荐系统的恶意用户检测

恶意用户攻击(托攻击, Shilling Attacks),托攻击者将伪造的用户模型注入推荐系统，这些伪造的用户模型通常能成为很多正常用户的近邻，从而扭曲系统的推荐结果

正常用户

目标项目

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation with Alice
Alice	5	2	3	3		?	
User1	2		4		4	1	1.00
User2	3	1	3		1	2	0.76
User3	4	2	3	1		1	0.72
User4	3	3	2	1	3	1	0.21
User5		3		1	2		-1.00
User6	4	3		3	3	2	0.94
User7		5		1	5	1	-1.00
Attack1	**5**		**3**		**2**	**5**	**1.00**
Attack2	**5**	**1**	**4**		**2**	**5**	**0.89**
Attack3	**5**	**2**	**2**	**2**		**5**	**0.93**

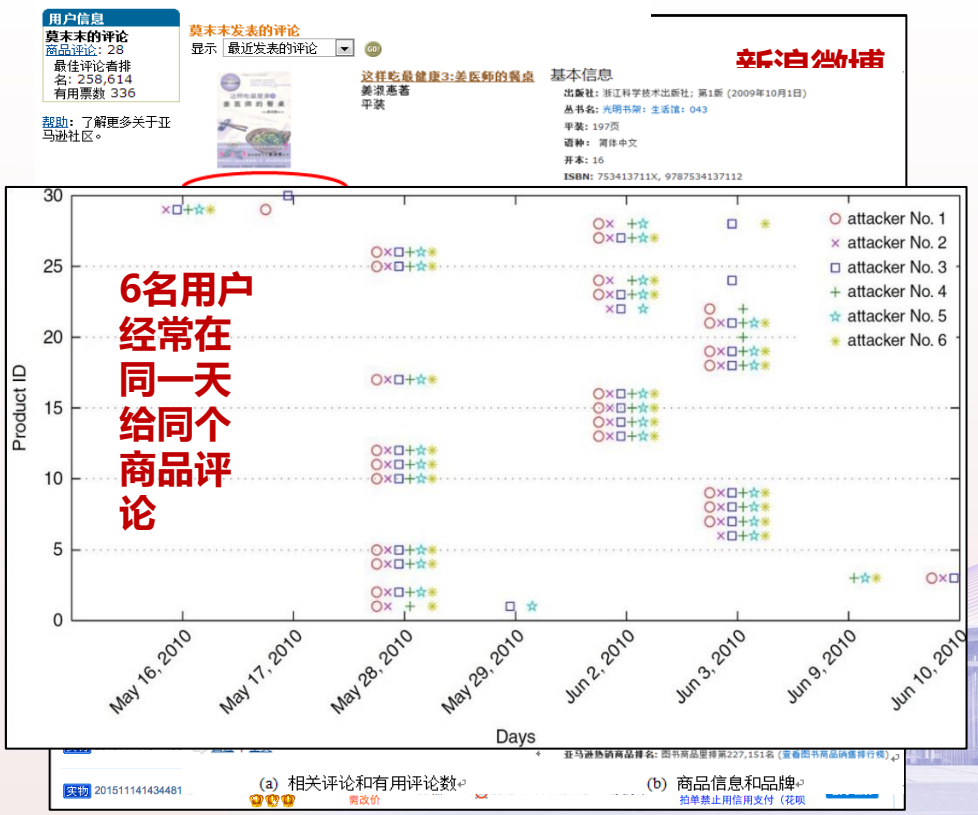
正常情况下，项目6未进入Alice的推荐列表

注入3个托攻击用户后，目标项目6推荐给了Alice

一、恶意用户检测简介

6.社会化商务恶意用户行为特征？

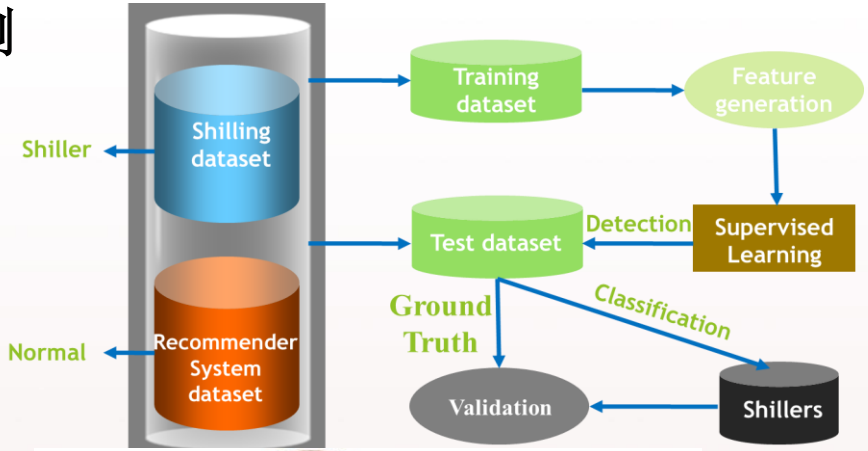
社会化商务恶意用户的目标通常为获得经济利益或造成网络影响，其行为模式必然与正常用户相比具有很大的差异性。



二、恶意用户检测方法

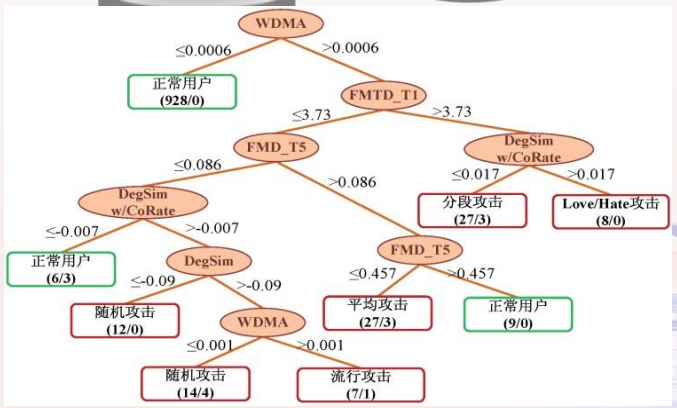
1.监督学习检测器

监督检测器将恶意用户检测看成二分类问题，并基于评分或评论特征来检测



朴素贝叶斯检测器

一个基于C4.5的推荐系统恶意用户检测示例



算法 4.2 基于朴素贝叶斯分类的恶意用户检测算法

输入：由少数正常用户和注入的恶意用户组成的训练数据集，以及测试数据集

输出：每个测试数据集中 u 的类别

1. 选取部分正常用户，并注入少部分恶意用户组成训练数据集
2. 计算训练和测试数据每个用户 u 的 n 个检测指标值，记为 $\{x_1, x_2, \ldots, x_n\}$
3. 利用 4.3.2 节中特征选择算法选择 m 个检测指标
4. 根据式(4.13)计算 $p(x_{ik}|c_j)$，μ_{jl} 和 σ_{jl} 分别为 C_j 类训练数据集第 l 个指标的均值和标准差：

$$p(x_{il}|c_j) = g(x_k, \mu_{jl}, \sigma_{jl}^2), \text{其中} g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (4.13)$$

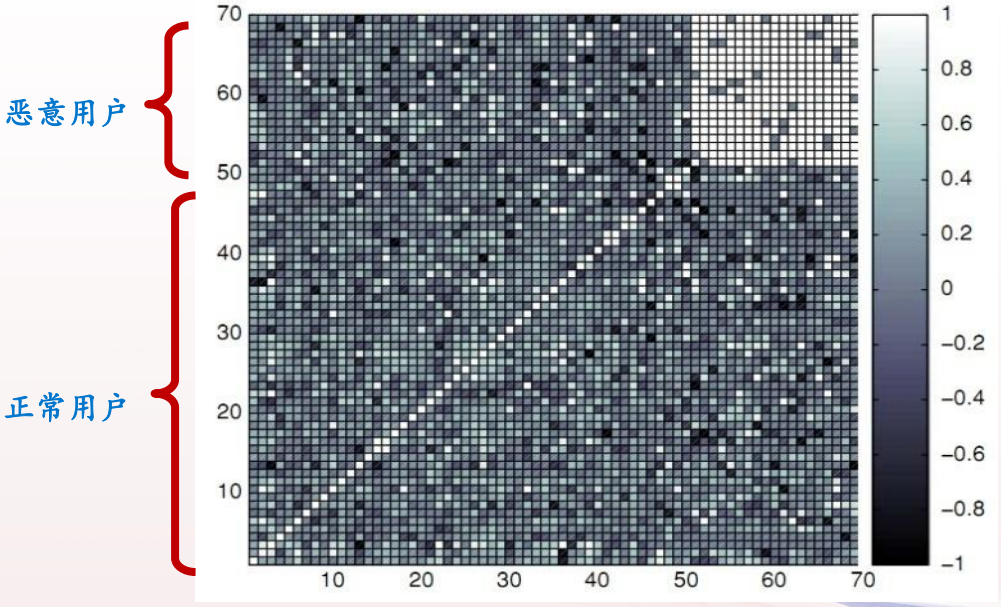
5. 根据公式(4.14)计算拖攻击属于正常或恶意用户的值：

$$\Omega = \ln\frac{p(S|u)}{p(N|u)} = \ln\frac{P(S)}{P(N)} + \sum_{k=1}^{m}\ln\frac{p(x_k|S)}{p(x_k|S)} \quad (4.14)$$

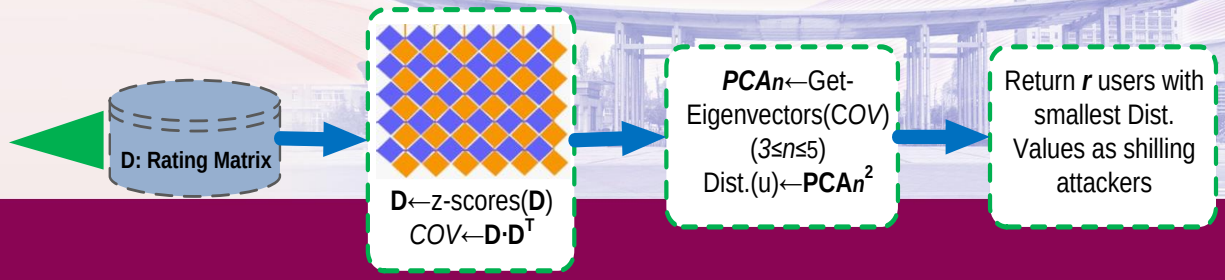
二、恶意用户检测方法

2. PCA检测器

但该算法直接基于用户评分矩阵，其前提条件为：恶意用户之间的PCC相似度非常高，而与正常用户之间的差异性则比较大



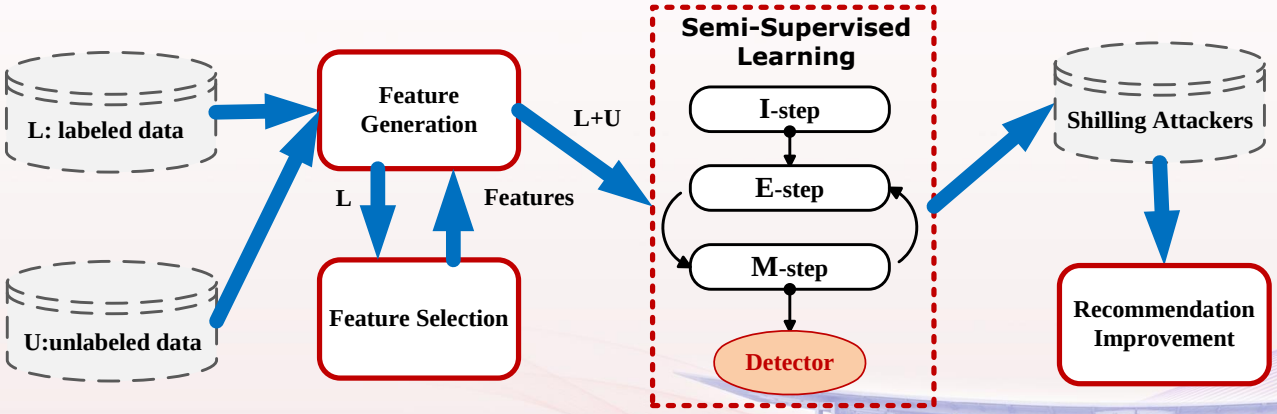
PCA检测器框架



二、恶意用户检测方法

3.半监督学习检测器

基于半监督学习的托攻击检测算法：
无标记数据往往容易获取，标记数据的获取往往费事费力。
怎样把有标记的数据和没有标记的数据结合起来，进行半监督的机器学习算法？
HySAD (Hybrid Shilling Attack Detector)



Wu Z, Wu J, Cao J, et al. HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation[C]//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2012: 985-993.

二、恶意用户检测方法

3.半监督学习检测器

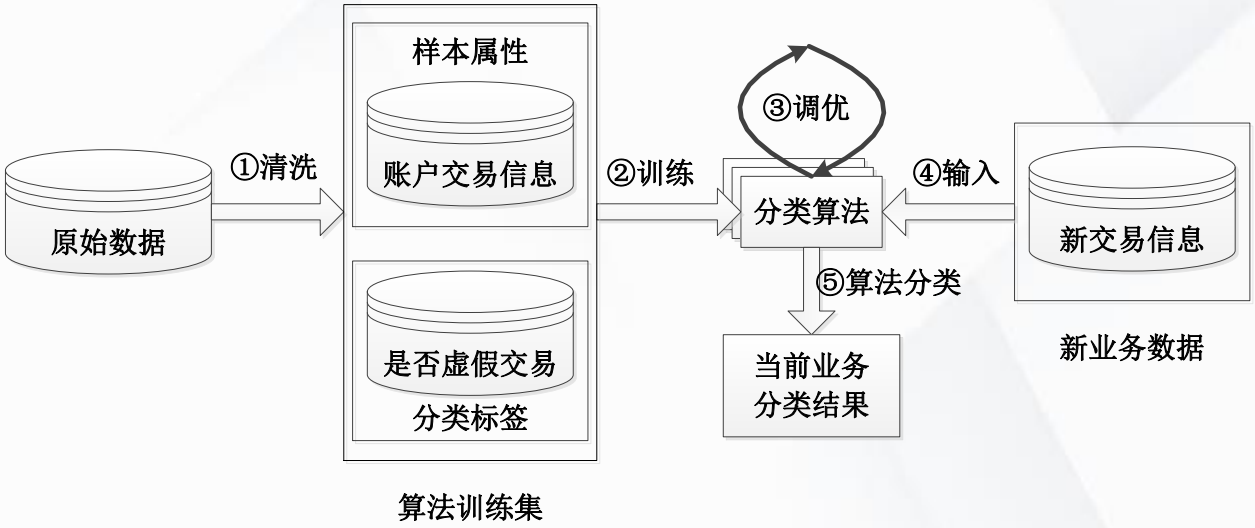
- **HySAD半监督学习训练过程可分为三个阶段**
  - **Initial Step:** 在标记数据集L上获得参数估计，与朴素贝叶斯过程一样
  - **E-step:** 得到无标记数据集U上的概率
  - **M-step:** 更新所有类的参数估计
- **几点说明**
  - 选用贝叶斯方法，假设参数服从正态分布仅仅是一种选择，可解释性比较好
  - 很好地融合了标记数据集的精确性，以及隐藏在无标记数据中的分布特性

01 案例背景

第一节 案例背景

- 信用卡虚假交易是指通过不存在的、伪造变更的实体商品或者服务交易，来套取信用卡内资金的违法行为。
- 进行虚假交易的持卡人往往缺乏正常健康的还款能力，因此会给信用卡发行机构的资金安全带来威胁。
- 同时，虚假交易套取的资金往往流向了限制或者禁止资金进入的领域，助长了违法犯罪活动的发生。
- 因此，对信用卡的虚假交易进行识别，是信用卡发行机构和各国政府执法部门密切关注的一项日常任务。
- 信用卡虚假交易识别，先后经历了人工甄别、规则判断和大数据人工智能识别等三个阶段。

- 目前，虚假交易识别已进入大数据人工智能识别阶段。该阶段借助信用卡交易大数据记录和飞速发展的数据挖掘算法，能在第一时间发现可疑的虚假交易，从而避免损失、遏制违法犯罪活动。
- 基于数据挖掘算法的虚假交易鉴别业务流程



- 第一步：是对原始业务记录数据进行清洗，得到满足算法输入要求的训练集数据。因为银行交易记录一般是非常规范和完善的，因此这一步的工作往往集中对数据项进行合理地数值化。账户的基本信息和每一笔交易信息，是算法训练集的属性；而对该笔交易进行人工分析或者事后反馈的结果，则是分类标签。
- 第二步：是选用某种数据挖掘算法进行模型训练。
- 第三步：是算法参数调优。
- 第四步：训练好的算法模型可以用来对第四步的新交易信息进行分类预测。
- 第五步：算法的输出即为预测的分类结果，可以及时地判定当前新交易是否有虚假交易的可能，从而采取相应的措施。

02

算法评价指标

第二节 算法评价指标

- 要评价一个数据挖掘算法效果的好坏，必须有可以量化的评价指标。
- 假设有一个数据集，一共有100个样本，其中包含60个逾期还款（阳性）样本，40个正常还款（阴性）样本。分类算法的目标是能够对正常和逾期还款的样本进行正确分类。假设某个分类算法将100个样本分成两类，每一类均为50个样本，其中分类到阳性逾期类别的样本中只有40个是真实的阳性逾期样本，另外10个样本属于阴性正常类别，即分类算法出现了错误分类，一方面将10个阴性样本误分为阳性，另一方面将20个阳性样本误分为阴性。

- 假设分类算法的预测类别分为阳性Positive和阴性Negative两类，预测结果的正确性用正正确（True）和错误(False)表示，则预测标签和数据真实标签之间的关系有以下四种：TP（正确预测的阳性逾期结果，即TruePositive）、TN（正确预测的阴性正常结果，即TrueNegative）、FP（错误预测的阳性逾期结果，即FalsePositive）和FN（错误预测的阴性正常结果，即FalseNegative）。按照这种表示方法，样本的分类情况可由表 15-1表示。

表 15-1 分类算法预测结果

		预测结果		总计
		逾期阳性	正常阴性	
实际类别	逾期阳性	TP=40	FN=20	P=60
	正常阴性	FP=10	TN=30	N=40
	总计	50	50	

分类模型的常见评价指标包括：

1. **准确率*Accuracy***：准确率是最直观的评价指标，即被正确分类的样本数除以样本总数。上面这个例子中正确率$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} = \frac{40+30}{100} = 0.7$。

2. **精确率*Precision***：即所有被分类为逾期阳性的样本中，实际上是逾期阳性的样本所占的比例。上面这个例子的精确率$Precision = \frac{TP}{TP+FP} = \frac{40}{40+10} = \frac{4}{5}$。

3. **召回率*Recall***：即所有真实分类为逾期阳性的样本中，被分类算法分类为逾期阳性的样本所占的比例。上面这个例子的召回率$Recall = \frac{TP}{TP+FN} = \frac{40}{40+20} = \frac{2}{3}$。

4. **假阳率*FalsePositiveRate***：该指标表示被错误预测为逾期阳性（意味着它实际值是正常阴性）的样本，占所有实际为正常阴性的比例。上面这个例子的假阳率$FPR = \frac{FP}{TN+FP} = \frac{10}{10+30} = \frac{1}{4}$。

5. **真阳率*TruePositiveRate***：该指标表示被正确预测为逾期阳性的样本，占所有实际为逾期阳性的比例。上面这个例子的真阳率和召回率*Recall*相等$TPR = \frac{TP}{TP+FN} = \frac{40}{40+20} = \frac{2}{3}$。

- 其他用来衡量分类算法性能的指标还包括：错误率（Error Rate = 1-Accracy）、灵敏度（Sensitive= TP/P）、特效度（Specificity= TN/N)以及算法计算速度、可扩展性、鲁棒性（算法处理异常值和缺失值的能力）和可解释性等。

混淆矩阵

- 在Python数据挖掘的应用中，很多算法训练后将得到表 15-1的结果报告，称之为混淆矩阵(Confusion Matrix)，进而计算得到上面提到的衡量指标。

算法训练、测试与评估

- 很多数据挖掘算法会将AUC的值和混淆矩阵一起输出，便于使用者对算法的分类效果进行评估。

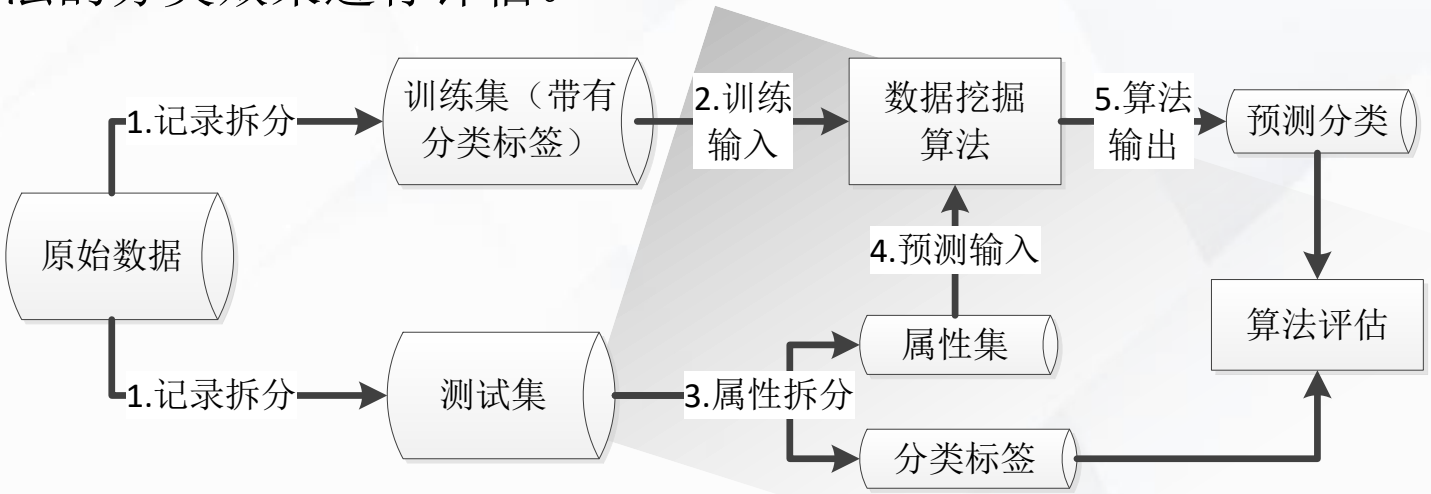


图 15-3 算法训练、测试与评估

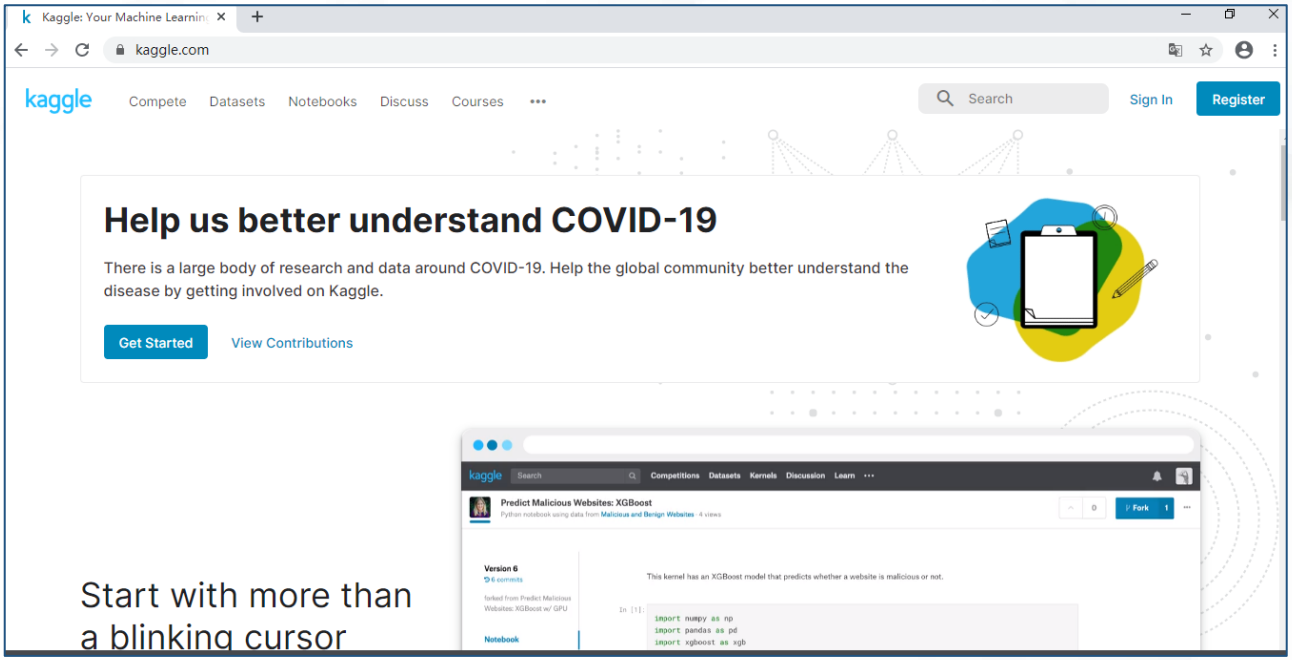
- 图 15-3描述了算法训练、测试与评估的主要过程。首先将原始数据拆分成训练集和测试集。
- 训练集用于数据挖掘算法模型的学习，从而得到模型的各项参数（即知识）。
- 测试集用于检验模型的分类效果，即将测试样本集输入到模型中，查看模型的输出是否与实际相符。
- 通过对预测分类结果和实际样本标签进行比较，就得到算法的评估情况。

- 在开展数据挖掘业务的过程中，模型接收的一般是一个静态数据集。
- 比方说我们的数据集是银行个人贷款的资料，其中包含了客户的个人信用信息、账户流水信息、以及客户是否逾期还款信息（即分类标签）。
- 这个数据集通常会分为两部分：训练集和测试集。
- 训练集可能占据整个数据集的80~90%，用来训练数据挖掘模型。
- 模型训练完后，使用测试集检验模型的分类预测情况，得到一个分类结果（不会逾期/会逾期）。
- 这个预测结果和测试集里的实际结果进行对比，就是表 15-1混淆矩阵中TP、FN、FP、TN数据。

03 数据概况

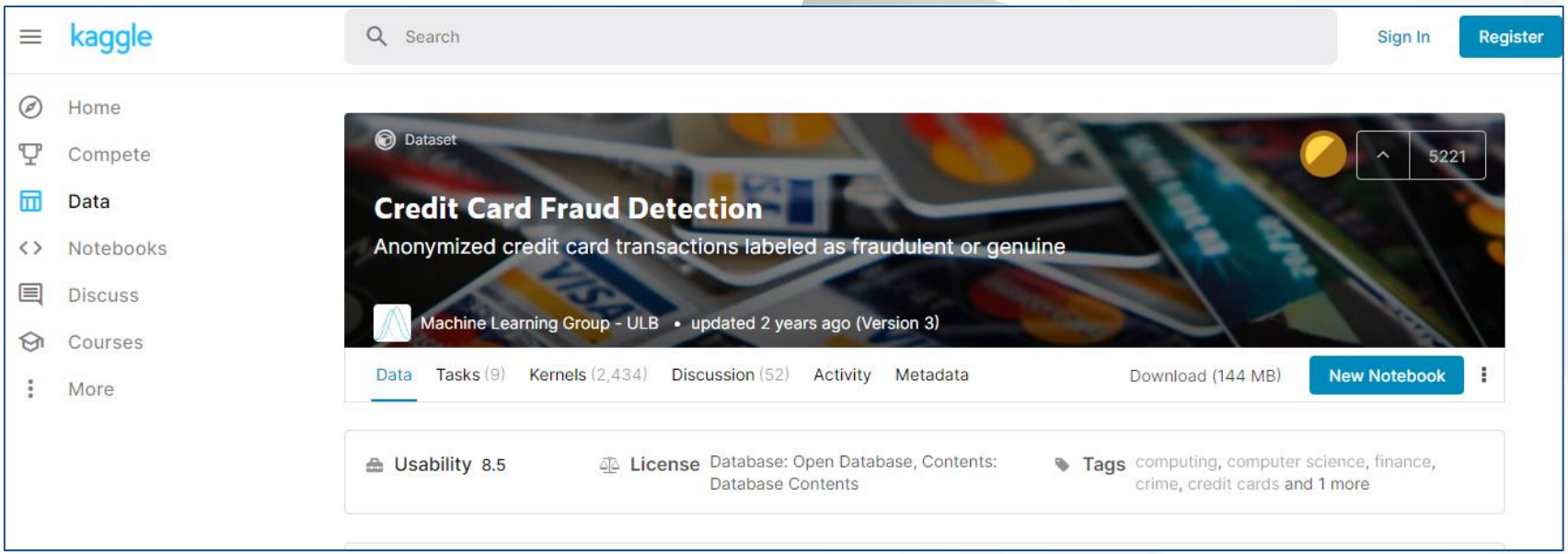
数据说明

- Kaggle网址：https://www.kaggle.com



第三节 数据概况

- 数据文件来自一个欧洲的信用卡发卡机构，记录了2013年9月中某两天的信用卡刷卡活动。
- 数据下载地址：https://www.kaggle.com/mlg-ulb/creditcardfraud。



数据说明

- 经过事后分析判断，在数据文件记录的284807次交易行为中，有492次虚假交易行为。
- 因为涉及到商业秘密和客户隐私，数据文件是原始记录经过主成分分析法PCA（从较多维度数据中提取出有价值的较少维度的数据，从而降低数据挖掘算法的计算难度）以后提取出来的。
- 一共有284807条记录，每条记录对应一次交易的持卡人信息或者交易信息。
- 每条记录由28个字段，用V1，V2……V28表示，除了交易时间和交易金额是原始数据之外，其他字段都经过了转换。“Class”属性取1，则表示是一个虚假交易；0则表示是正常交易。

04 操作流程

一、数据清洗与因素分析

- 首先，准备好所需的Python工具包：NumPy、Pandas和Matplotlib等三个基础包；支持数据挖掘算法还的Sklearn包；用于统计数据可视化的seaborn包等。下列代码主要用于加载程序中要用到的工具包。

```
"""
以下代码参考了 https://www.kaggle.com/mlg-ulb/creditcardfraud 的若干案例
向原作者致谢!
"""
#工作包准备
print('*'*45)
print('工作包准备:')
print('*'*45)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#中文图表显示字体以及坐标轴负号显示设置
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

import matplotlib.gridspec as gridspec
import seaborn as sns;
plt.style.use('ggplot')

import sklearn
#预处理，对数据做均值和方差归一
from sklearn.preprocessing import StandardScaler
#用于将整个数据集划分成训练集和测试集
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
#用于输出混淆矩阵
from sklearn.metrics import confusion_matrix
```

- 对于实际的大数据处理和数据挖掘应用案例来说，数据准备阶段占据的比重很大，也非常重要。下列代码主要用来读取数据，并展示数据的基本情况。读入交易数据，显示数据概况：

```
#数据概况
print()
print('*'*45)
print('读入交易数据，显示数据概况:')
print('*'*45)
#crecreditcard_data=pd.read_csv('./creditcard.csv')
FraudDetction=pd.read_csv('../../Examples/FraudCredit/creditcard.csv')
FraudDetction.info()
FraudDetction.describe()

```

```
# 看看欺诈交易与正常交易的数据量对比
label_distr=pd.value_counts(FraudDetction['Class'],sort=True).sort_index()
# 统计欺诈与正常交易的总数
label_distr.value_counts()
#绘制饼图
label_distr.plot.pie(legend='True',colors=['b','w'],
                     title='正常与欺诈用户分布\n0:正常  1:欺诈')
plt.savefig('ch17_01.png',dpi=300,bbox_inches='tight')
plt.show()
```

- 输出：

```
Out: ********************************************
     读入交易数据，显示数据概况:
     ********************************************
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 284807 entries, 0 to 284806
     Data columns (total 31 columns):
     Time        284807 non-null float64
     V1          284807 non-null float64
     V2          284807 non-null float64
     V3          284807 non-null float64
     ......
     V28         284807 non-null float64
     Amount      284807 non-null float64
     Class       284807 non-null int64
     dtypes: float64(30), int64(1)
     memory usage: 67.4 MB
```

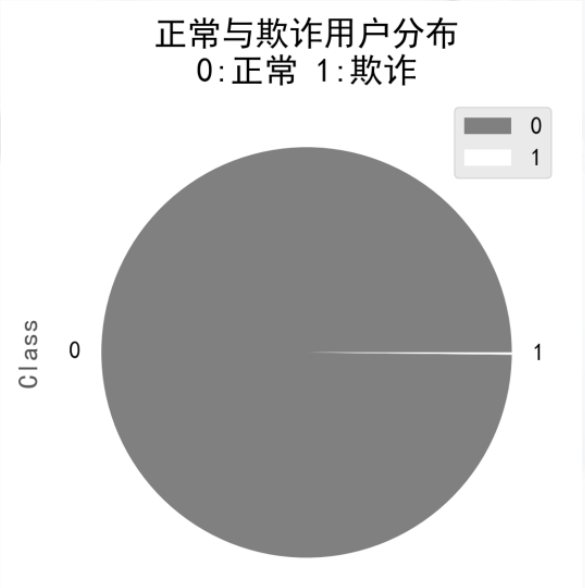


图 15-4 正常交易与欺诈交易饼图

可以看出，原始数据包括Time，V1,V2……V28，Amount，Class等列，共有284807行，没有缺失值。如果按照Class的取值分类绘制饼图，可以发现正常交易占了绝大多数比重。按照原始数据文件说明，虚假交易的比例是492/284807=0.1727%。

- 接下来提出几个猜想，例如虚假交易是否和交易金融有关？是否和交易时间有关？然后验证这些猜想。

```
#查看正常/欺诈交易与交易时间的关系
print('*'*45)
print('查看正常/欺诈交易发生数量与交易时间的关系:')
print('*'*45)

# 查看二者的描述性统计,与时间的序列分布关系
print('正常')
print(FraudDetction.Time[FraudDetction.Class == 0].describe())
print('-'*25)
print('欺诈')
print(FraudDetction.Time[FraudDetction.Class == 1].describe())

f,(ax1,ax2)=plt.subplots(2,1,sharex=True,figsize=(12,6))
bins=100

ax1.hist(FraudDetction.Time[FraudDetction.Class == 1],bins=bins)
ax1.set_title('欺诈(Fraud))',fontsize=22)
ax1.set_ylabel('交易量',fontsize=15)

ax2.hist(FraudDetction.Time[FraudDetction.Class == 0],bins=bins)
ax2.set_title('正常(Normal）',fontsize=22)
ax2.set_ylabel('交易量',fontsize=15)

plt.xlabel('时间(单位:秒)',fontsize=15)
plt.xticks(fontsize=15)

print()
print('交易量与时间的关系：')
print()
plt.savefig('ch17_02.png',dpi=300,bbox_inches='tight')
plt.show()
```

- 输出结果：

```
Out: ********************************************
     查看正常/欺诈交易发生数量与交易时间的关系:
     ********************************************
     正常
     count    284315.000000
     mean      94838.202258
     std       47484.015786
     min           0.000000
     25%       54230.000000
     50%       84711.000000
     75%      139333.000000
     max      172792.000000
     Name: Time, dtype: float64
     ------------------------
     欺诈
     count       492.000000
     mean      80746.806911
     std       47835.365138
     min         406.000000
     25%       41241.500000
     50%       75568.500000
     75%      128483.000000
     max      170348.000000
     Name: Time, dtype: float64
```

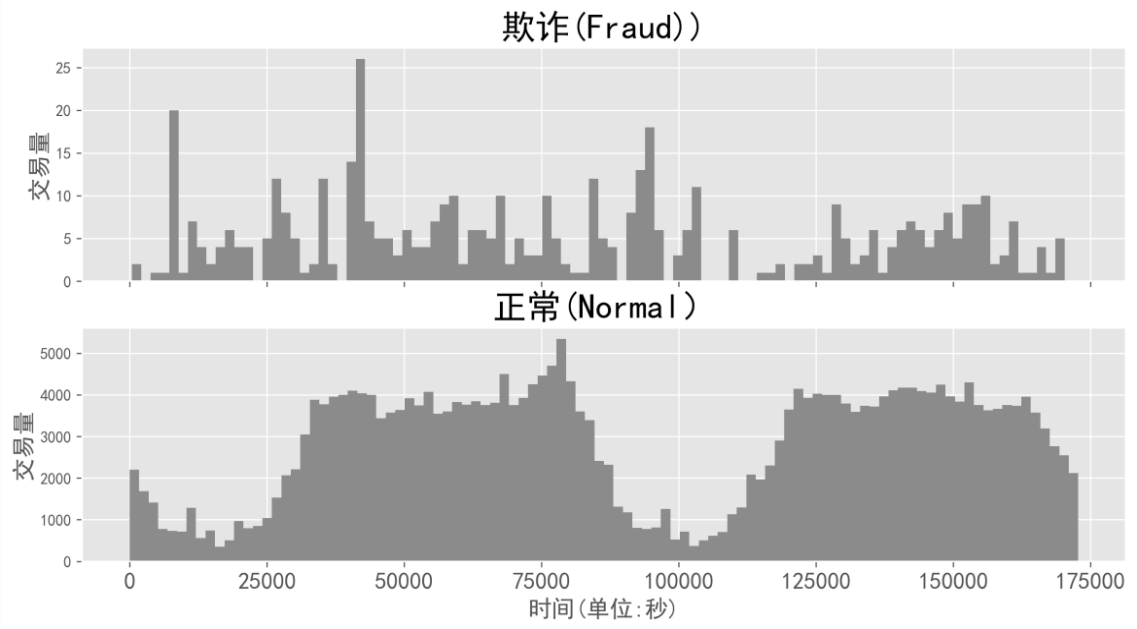


图 15-5 交易量与时间的关系

- 真实/欺诈交易与交易金额的关系

```
#查看真实/欺诈交易与交易金额的关系
print()
print('*'*45)
print("查看正常/欺诈交易发生数量与单笔交易金额的关系:")
print('*'*45)
# 查看二者的描述性统计,与金额的序列分布关系
print('正常')
print(FraudDetction.Amount[FraudDetction.Class==0].describe())
print('-'*25)
print('欺诈')
print(FraudDetction.Amount[FraudDetction.Class ==1].describe())

f,(ax1,ax2)=plt.subplots(2,1,sharex=False,figsize=(12,6))
bins=100

ax1.hist(FraudDetction.Amount[FraudDetction.Class == 1],bins=bins)
ax1.set_title('欺诈(Fraud)',fontsize=22)
ax1.set_ylabel('交易量',fontsize=15)

ax2.hist(FraudDetction.Amount[FraudDetction.Class == 0],bins=bins)
ax2.set_title('正常(Normal)',fontsize=22)
ax2.set_ylabel('交易量',fontsize=15)

plt.xlabel('金额($)',fontsize=15)
plt.xticks(fontsize=15)
#因为金额数据跨度过大，因此 y 轴调整成对数坐标
plt.yscale('log')

print()
print('交易量与交易金额的关系：')
plt.savefig('ch17_03.png',dpi=300,bbox_inches='tight')
plt.show()
```

- 输出结果：

```
Out: ********************************************
     查看正常/欺诈交易发生数量与单笔交易金额的关系:
     ********************************************
     正常
     count      284315.000000
     mean           88.291022
     std           250.105092
     min             0.000000
     25%             5.650000
     50%            22.000000
     75%            77.050000
     max         25691.160000
     Name: Amount, dtype: float64
     ------------------------
     欺诈
     count        492.000000
     mean         122.211321
     std          256.683288
     min            0.000000
     25%            1.000000
     50%            9.250000
     75%          105.890000
     max         2125.870000
     Name: Amount, dtype: float64
```

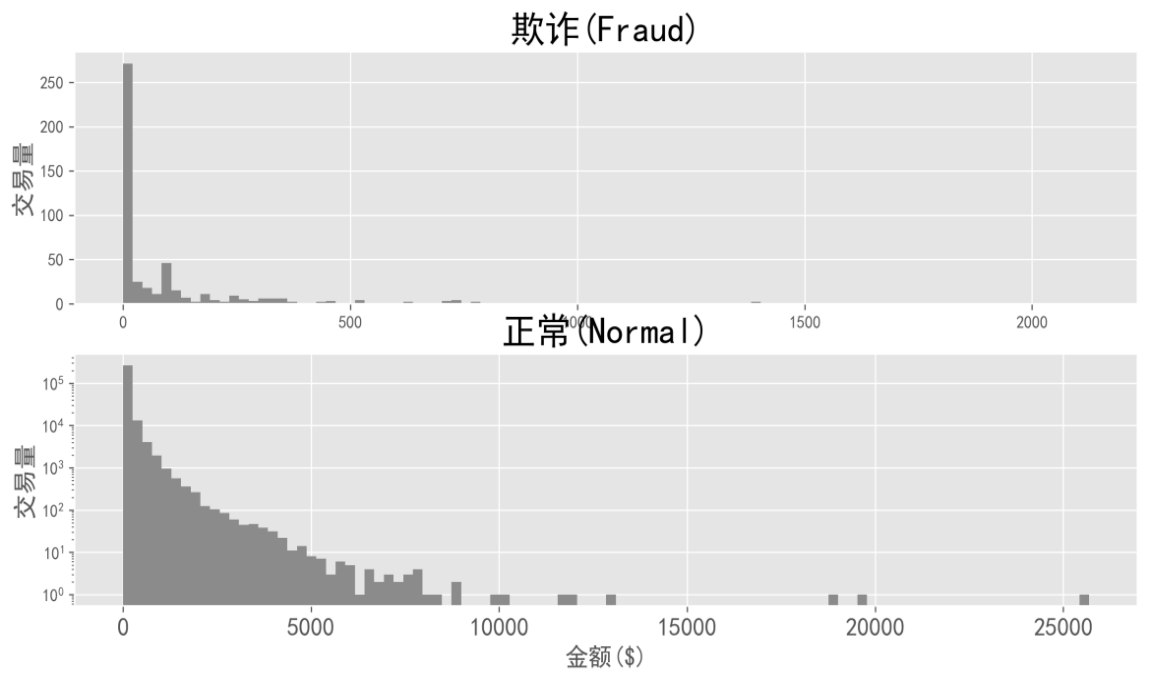


图 15-6 交易量与金额的关系

从图 15-5和图 15-6来看，虚假交易和正常交易在交易时间和交易金额这两个维度上都没有表现出明显的不一致。也就意味从交易时间和交易金额上不能有效区分虚假和真实交易。

- 接下来对V1-V28个属性进行逐个验证，看看在哪一个属性上，虚假交易和真实交易有比较明显的区别。

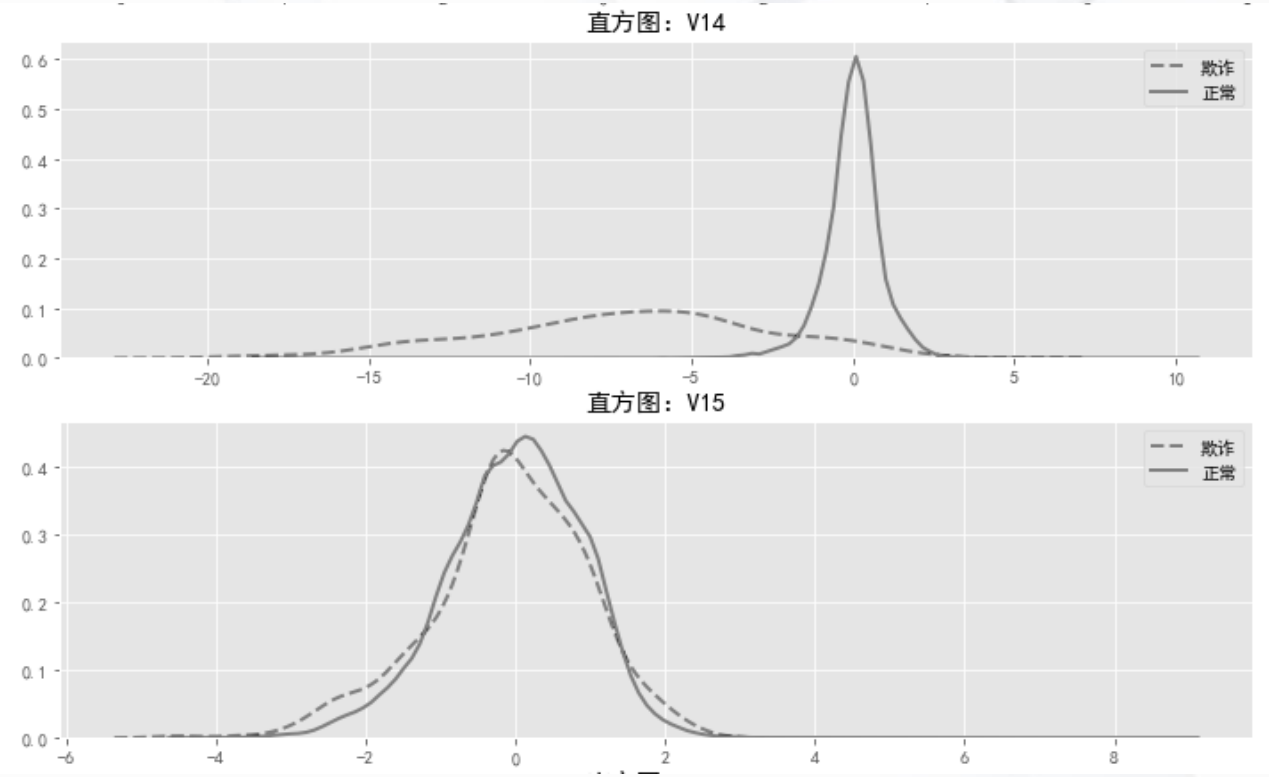
```
#分别查看 28 个属性与正常/欺诈的分类标签之间的联系是否紧密
#两个曲线越是趋向一致，说明该属性越难以区分正常或者欺诈交易
#两个曲线越是差别大，说明该属性的取值对正常/欺诈交易的区分越明显
#下列代码输出结果中的“直方图 V14”，正常/欺诈交易的分布区别很明显
#而“直方图 V15”，正常/欺诈交易的分布就基本重合
print()
print('*'*45)
print('查看正常/欺诈交易在各属性下的分布关系:')
print('*'*45)
v_features=[x for x in FraudDetction.columns
                    if x not in ['Time','Amount','Class']]
plt.figure(figsize=(12,28*4))
gs =gridspec.GridSpec(28,1)

```

```
import warnings
warnings.filterwarnings('ignore')

for i,cn in enumerate(FraudDetction[v_features]):
    ax=plt.subplot(gs[i])
    sns.distplot(FraudDetction[cn][FraudDetction.Class==1],
                 hist=False,bins=50,kde=True,label="欺诈",
                 kde_kws={"color": "r", "lw": 2, 'linestyle':'--'},)
    sns.distplot(FraudDetction[cn][FraudDetction.Class==0],
                 hist=False,bins=50,kde=True,label="正常",
                 kde_kws={"color": "b", "lw": 2, 'linestyle':'-'},)
    ax.set_xlabel('')
    ax.set_title('直方图：'+str(cn))
plt.savefig('ch17_04.png',dpi=300,bbox_inches='tight')
plt.show()
```

- 输出：正常/欺诈交易在各属性下的分布关系



可以看到，在属性V14上，虚假交易和正常交易的分布区分比较明显；而在属性V15上，虚假交易和正常交易是叠加在一起的，很难区分。如果要做进一步的相关性分析的话，V14和V15会有截然不同的结论。

图 15-7

- 很明显，如果只选取部分属性作为欺诈交易监测的目标，那么应该重点关注像V14这样的属性。因为在这个属性上，正常交易和欺诈交易有明显的差异。在实际应用中，是否要进行属性选择，取决于业务特征、数据规模、运算资源、决策时效等因素。本案例后续的分析中，选择了全部属性参与模型训练。
- 接下来使用该数据集来训练Sklearn内置的逻辑回归、随机森林和支持向量机等三种算法模型。

二、模型训练

- 首先要将数据集分成训练集和测试集。

```
#分别调用逻辑回归、随机森林、支持向量 SVM 三种 Scikit 内建的模型
#数据按 7：3 的比例生产训练和测试数据集分组
print()
print('*'*45)
print('模型训练数据准备:')
print('*'*45)
Fraud=FraudDetction[FraudDetction.Class == 1]
Normal=FraudDetction[FraudDetction.Class == 0]

# 训练特征集
x_train=Fraud.sample(frac=0.7)
x_train=pd.concat([x_train,Normal.sample(frac=0.7)],axis=0)
# 测试特征集
x_test=FraudDetction.loc[~FraudDetction.index.isin(x_train.index)]
```

```

# 标签集
y_train=x_train.Class
y_test=x_test.Class

# 去掉特征集里的标签和时间列
x_train=x_train.drop(['Class','Time'],axis=1)
x_test=x_test.drop(['Class','Time'],axis=1)
# 查看数据结构
print('训练数据结构：',x_train.shape,y_train.shape)
print('测试数据结构：',x_test.shape,y_test.shape)

#导入日期时间包，用于算法训练时间分析
import datetime
```

- 将70%的数据作为训练集，30%作为测试集数据，也可以按照75%-25%分拆。
- 使用数据集依次对三种算法进行训练与测试。需要强调的是：每次运行的最终结果可能会有一些微小的差异，这主要是由样本的随机抽样结果和算法本身的数学特性导致的。

```
Out:  **********************************************
      模型训练数据准备:
      **********************************************
      训练数据结构：  (199364, 29) (199364,)
      测试数据结构：  (85443, 29) (85443,)
```

1. 逻辑回归方法

```
#逻辑回归方法
print()
print('*'*45)
print('逻辑回归')
print('*'*45)
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (precision_recall_curve,auc,roc_auc_score,
roc_curve,recall_score,classification_report)

#构造逻辑回归模型
lrmodel = LogisticRegression(penalty='l2')
```

```
#训练模型
start=datetime.datetime.now()
lrmodel.fit(x_train, y_train)
end=datetime.datetime.now()

#查看混淆矩阵
ypred_lr=lrmodel.predict(x_test)
print('confusion_matrix')
print(metrics.confusion_matrix(y_test,ypred_lr))

#查看预测精度与决策覆盖面
print('Accuracy:%f'%(metrics.accuracy_score(y_test,ypred_lr)))
print('Area under the curve:%f'%(metrics.roc_auc_score(y_test,ypred_lr)))
print('Runtime =',end-start)
```

- 输出结果：

```
Out:  ****************************
      逻辑回归
      ****************************
      confusion_matrix
      [[85283       12]
       [    64       84]]
      Accuracy:0.999111
      Area under the curve:0.783713
      Runtime = 0:00:07.291813
```

从逻辑回归的结果来看，在85443个测试样本中，有85283个正常交易样本被识别，84个虚假交易样本被识别。但是，也有12个正常样本被错误标识成虚假交易，而64个虚假交易样本被错误标识成正常。

2. 随机森林算法

- 以决策树为基础，随机森林算法将多棵树集成为一片森林：每棵决策树都是一个分类器（假设现在针对的是分类问题），那么对于一个输入样本，N棵树会有N个分类结果。
- 而随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出。
- 因为随机森林算法是一个统计投票结果，因此那些随机扰动产生的错误会在很大程度上被缓解，其结果会更加稳定，准确率也比较高。Scikit中已经集成了随机森林算法，可以直接调用。

- 随机森林算法程序如下：

```
#随机森林模型
print()
print('*'*45)
print('随机森林')
print('*'*45)
from sklearn.ensemble import RandomForestClassifier

#构造随机森林模型
rfmodel=RandomForestClassifier()
#训练模型
start=datetime.datetime.now()
rfmodel.fit(x_train,y_train)
end=datetime.datetime.now()

#查看混淆矩阵
ypred_rf=rfmodel.predict(x_test)
print('confusion_matrix')
print(metrics.confusion_matrix(y_test,ypred_rf))

#查看预测精度与决策覆盖面
print('Accuracy:%f'%(metrics.accuracy_score(y_test,ypred_rf)))
print('Area under the curve:%f'%(metrics.roc_auc_score(y_test,ypred_rf)))
print('Runtime =',end-start)
```

- 输出结果：

```
Out:  ************************************************
      随机森林
      ************************************************
      confusion_matrix
      [[85284      11]
       [    31    117]]
      Accuracy:0.999508
      Area under the curve:0.895206
      Runtime = 0:00:12.279130
```

从随机森林的结果来看，在 85443 个测试样本中，有 85284 个正常交易样本被识别，117 个虚假交易样本被识别。有 11 个正常样本被错误标识成虚假交易，而 31 个虚假样本被错误标识成正常。

3. 支持向量机算法

```
#支持向量机 SVM
print()
print('*'*45)
print('支持向量机 SVM')
print('*'*45)
from sklearn.svm import SVC

#构造支持向量机模型
svcmodel=SVC(kernel='sigmoid')
#训练模型
start=datetime.datetime.now()
svcmodel.fit(x_train,y_train)
end=datetime.datetime.now()
```

```

#查看混淆矩阵
ypred_svc=svcmodel.predict(x_test)
print('confusion_matrix')
print(metrics.confusion_matrix(y_test,ypred_svc))

#查看预测精度与决策覆盖面
print('Accuracy:%f'%(metrics.accuracy_score(y_test,ypred_svc)))
print('Area under the curve:%f'%(metrics.roc_auc_score(y_test,ypred_svc)))
print('Runtime =',end-start)
```

- 输出结果：

```
Out:  ****************************
      支持向量机 SVM
      ****************************
      confusion_matrix
      [[85163     132]
       [   138      10]]
      Accuracy:0.996840
      Area under the curve:0.533010
      Runtime = 0:00:09.024050
```

从支持向量机的结果来看，在85443个测试样本中，有85163个正常交易样本被识别，只有10个虚假交易样本被识别。有132个正常样本被错误标识成虚假，而138个虚假样本被错误标识成正常。

- 在这个案例中，预测效果最好的是随机森林模型；运行速度最快的是逻辑回归；
- 而支持向量机的表现比较差，大部分的虚假交易样本都没有被检测出来，同时算法标识的虚假交易中有很多被错误标识的正常交易。
- 可见，在真实的案例中，并不是数学理论复杂的方法一定会有好的效果。
- 本例的三个分类算法中，随机森林算法相对较简单，但在预测效果和执行速度上，都取得了更好的结果。

三、模型应用

- 模型训练好以后，就可以使用模型进行预测了。从x_test测试集中随机抽取一条记录，考察用三种训练好的模型对它进行正常/欺诈交易分类的结果。

```
#使用训练好的随机森林模型进行预测
#从 x_test 中抽取一条记录，作为模拟的新输入数据
new_input=x_test.iloc[10].values.reshape(1,-1)
print("new_input=",new_input)
#使用逻辑回归模型对该数据进行预测
print("lrmodel prediction=",lrmodel.predict(new_input))
#使用随机森林模型对该数据进行预测
print("rfmodel prediction=",rfmodel.predict(new_input))
#使用支持向量机模型对该数据进行预测
print("SVCmodel prediction=",svcmodel.predict(new_input))
```

- 输出结果：

```
Out:    new_input= [[-2.46045949e-01   4.73266903e-01   1.69573755e+00   2.62411488e-01
          -1.08664137e-02  -6.10835935e-01   7.93936546e-01  -2.47252831e-01
           1.38879114e-01  -4.01007068e-01  -8.12050382e-01  -1.83524462e-01
          -6.30112887e-01  -2.86217451e-01  -3.37046324e-01  -4.28105474e-01
          -3.06021083e-02  -5.04568166e-01   3.55180278e-01   4.57211273e-02
          -1.94599935e-01  -3.35132726e-01  -7.81796063e-02   3.92783892e-01
          -3.11571468e-02   1.98036799e-01  -1.75347301e-01  -2.03616156e-01
           3.04900000e+01]]
        lrmodel prediction= [0]
        rfmodel prediction= [0]
        SVCmodel prediction= [0]
```

明显地，对于输入的这样一组由V1,V2……V28项数据构成的一个样本，三个模型都给出相同的结果0，也就意味着这是一个真实的信用卡交易。
请注意，因为x_test测试集由随机抽取产生，所以每次运行取第10条得到的new_input很可能是不同的，当然，测试结果也会有差异。

谢谢观赏 下节课见