手机用户分类模型

摘 要

随着人们使用手机的频率越来越高，电信运营商掌握着大量的手机用户的使用数据。如何充分利用这些海量的用户数据，从中挖掘出有价值的知识，是当前电信运营商们面临的一个重要课题。有了这些知识，电信运营商可以采取更好的商业策略，也可以对行为异常的用户进行预测。本文使用数据挖掘和机器学习的方法，采取常见的分类和异常检测的算法对用户数据进行分析，预测异常用户，从而对他们采取相应的措施。

关键词：机器学习，大数据，数据挖掘，分类算法，异常检测

### Models for mobile users classification

# **Abstract**

As people use mobile phones more and more frequently, telecom operators hold a lot of data on the usage of mobile phone users. How to fully make use of the vast amount of subscriber data and extract valuable knowledge from it is an important topic facing telecom operators nowadays. With this knowledge, telecom operators can adopt better business strategies and also make predictions about users with abnormal behavior. This paper uses classification and anomaly detection algorithms of data mining and machine learning to analyze user data and then predict anomalous users so that action can be taken to against them.

**Keywords**：Data Mining；Machine Learning；Big Data;Classification algorithm;anomaly detection

目录

[1.绪论](#_Toc513578580_WPSOffice_Level1) [5](#_Toc513578580_WPSOffice_Level1)

[1.1 研究背景](#_Toc1672954323_WPSOffice_Level2) [5](#_Toc1672954323_WPSOffice_Level2)

[1.2 相关理论](#_Toc1377732896_WPSOffice_Level2) [5](#_Toc1377732896_WPSOffice_Level2)

[1.2.1 分类算法](#_Toc1672954323_WPSOffice_Level3) [5](#_Toc1672954323_WPSOffice_Level3)

[1.2.2 异常检测模型](#_Toc1377732896_WPSOffice_Level3) [6](#_Toc1377732896_WPSOffice_Level3)

[1.3 数据轮廓](#_Toc1392384550_WPSOffice_Level2) [7](#_Toc1392384550_WPSOffice_Level2)

[1.4 系统开发](#_Toc2125175334_WPSOffice_Level2) [7](#_Toc2125175334_WPSOffice_Level2)

[1.4.1 硬件参数](#_Toc1392384550_WPSOffice_Level3) [7](#_Toc1392384550_WPSOffice_Level3)

[1.4.2 开发技术](#_Toc2125175334_WPSOffice_Level3) [7](#_Toc2125175334_WPSOffice_Level3)

[1.4.3 系统架构](#_Toc241269522_WPSOffice_Level3) [8](#_Toc241269522_WPSOffice_Level3)

[2. 数据预处理](#_Toc1672954323_WPSOffice_Level1) [8](#_Toc1672954323_WPSOffice_Level1)

[2.1 数据清洗](#_Toc241269522_WPSOffice_Level2) [8](#_Toc241269522_WPSOffice_Level2)

[2.1.1 处理缺失数据](#_Toc1040647645_WPSOffice_Level3) [8](#_Toc1040647645_WPSOffice_Level3)

[2.1.2 离群点检测](#_Toc1171030834_WPSOffice_Level3) [10](#_Toc1171030834_WPSOffice_Level3)

[2.2 数据无量纲化](#_Toc1040647645_WPSOffice_Level2) [11](#_Toc1040647645_WPSOffice_Level2)

[3. 特征工程](#_Toc1377732896_WPSOffice_Level1) [12](#_Toc1377732896_WPSOffice_Level1)

[(1) 特征提取(feature extraction)](#_Toc1171030834_WPSOffice_Level2) [12](#_Toc1171030834_WPSOffice_Level2)

[(2) 特征创造(feature creation)](#_Toc400635275_WPSOffice_Level2) [12](#_Toc400635275_WPSOffice_Level2)

[(3) 特征选择(feature selection)](#_Toc51807504_WPSOffice_Level2) [12](#_Toc51807504_WPSOffice_Level2)

[3.1 特征选择](#_Toc1688913027_WPSOffice_Level2) [13](#_Toc1688913027_WPSOffice_Level2)

[3.1.1 方差过滤](#_Toc400635275_WPSOffice_Level3) [13](#_Toc400635275_WPSOffice_Level3)

[3.1.2 卡方检验](#_Toc51807504_WPSOffice_Level3) [15](#_Toc51807504_WPSOffice_Level3)

[3.1.3 互信息法](#_Toc1688913027_WPSOffice_Level3) [17](#_Toc1688913027_WPSOffice_Level3)

[3.2 降维](#_Toc944084283_WPSOffice_Level2) [19](#_Toc944084283_WPSOffice_Level2)

[3.2.1 主成分分析](#_Toc944084283_WPSOffice_Level3) [19](#_Toc944084283_WPSOffice_Level3)

[3.2.2 线性判别分析](#_Toc1322194678_WPSOffice_Level3) [20](#_Toc1322194678_WPSOffice_Level3)

[4. 模型选择与评估](#_Toc1392384550_WPSOffice_Level1) [22](#_Toc1392384550_WPSOffice_Level1)

[4.1 交叉验证](#_Toc1322194678_WPSOffice_Level2) [22](#_Toc1322194678_WPSOffice_Level2)

[(1) 将原始数据平均分为k份。](#_Toc1842434175_WPSOffice_Level3) [23](#_Toc1842434175_WPSOffice_Level3)

[(2) 选一份数据做测试集来评估模型的表现，其余部分做训练集。](#_Toc1529270725_WPSOffice_Level3) [23](#_Toc1529270725_WPSOffice_Level3)

[(3) 重复第(2)步直到k份数据都被选中过做测试集。](#_Toc1457489268_WPSOffice_Level3) [23](#_Toc1457489268_WPSOffice_Level3)

[(4) 计算每一步模型表现的平均值。](#_Toc1191701776_WPSOffice_Level3) [23](#_Toc1191701776_WPSOffice_Level3)

[4.2 模型评估](#_Toc1842434175_WPSOffice_Level2) [23](#_Toc1842434175_WPSOffice_Level2)

[(1) 混淆矩阵](#_Toc1380267637_WPSOffice_Level3) [23](#_Toc1380267637_WPSOffice_Level3)

[(2) ROC曲线](#_Toc583137735_WPSOffice_Level3) [23](#_Toc583137735_WPSOffice_Level3)

[(3) AUC面积](#_Toc1389273165_WPSOffice_Level3) [23](#_Toc1389273165_WPSOffice_Level3)

[4.2.1 混淆矩阵](#_Toc1029741755_WPSOffice_Level3) [23](#_Toc1029741755_WPSOffice_Level3)

[4.2.2 ROC曲线与AUC面积](#_Toc944505697_WPSOffice_Level3) [24](#_Toc944505697_WPSOffice_Level3)

[5. 不均衡数据处理](#_Toc2125175334_WPSOffice_Level1) [25](#_Toc2125175334_WPSOffice_Level1)

[5.1 数据采样](#_Toc1529270725_WPSOffice_Level2) [26](#_Toc1529270725_WPSOffice_Level2)

[5.1.1 随机下采样](#_Toc1020101535_WPSOffice_Level3) [26](#_Toc1020101535_WPSOffice_Level3)

[5.1.2 Tomek Link方法](#_Toc940336763_WPSOffice_Level3) [26](#_Toc940336763_WPSOffice_Level3)

[5.1.3 NearMiss方法[]](#_Toc1570027840_WPSOffice_Level3) [26](#_Toc1570027840_WPSOffice_Level3)

[5.1.4 随机上采样](#_Toc1518017772_WPSOffice_Level3) [26](#_Toc1518017772_WPSOffice_Level3)

[5.1.5 SMOTE算法[]](#_Toc1410385241_WPSOffice_Level3) [27](#_Toc1410385241_WPSOffice_Level3)

[5.1.6 综合采样](#_Toc636886790_WPSOffice_Level3) [27](#_Toc636886790_WPSOffice_Level3)

[5.1.7 总结](#_Toc211198067_WPSOffice_Level3) [28](#_Toc211198067_WPSOffice_Level3)

[5.2 代价敏感学习](#_Toc1457489268_WPSOffice_Level2) [28](#_Toc1457489268_WPSOffice_Level2)

[6. 实验结果](#_Toc241269522_WPSOffice_Level1) [29](#_Toc241269522_WPSOffice_Level1)

[6.1 分类模型](#_Toc1191701776_WPSOffice_Level2) [29](#_Toc1191701776_WPSOffice_Level2)

[6.1.1 决策树](#_Toc882478532_WPSOffice_Level3) [29](#_Toc882478532_WPSOffice_Level3)

[6.1.1-3 CART模型运行效率](#_Toc1992472636_WPSOffice_Level3) [32](#_Toc1992472636_WPSOffice_Level3)

[6.1.2 随机森林](#_Toc111101778_WPSOffice_Level3) [32](#_Toc111101778_WPSOffice_Level3)

[6.1.3随机梯度下降](#_Toc407949207_WPSOffice_Level3) [34](#_Toc407949207_WPSOffice_Level3)

[6.1.4 支持向量机](#_Toc1222721885_WPSOffice_Level3) [35](#_Toc1222721885_WPSOffice_Level3)

[6.1.5 总结](#_Toc1503486328_WPSOffice_Level3) [35](#_Toc1503486328_WPSOffice_Level3)

[6.2 异常检测模型](#_Toc1380267637_WPSOffice_Level2) [39](#_Toc1380267637_WPSOffice_Level2)

[7. 总结](#_Toc1040647645_WPSOffice_Level1) [40](#_Toc1040647645_WPSOffice_Level1)

[8. 致谢](#_Toc1171030834_WPSOffice_Level1) [41](#_Toc1171030834_WPSOffice_Level1)

[参考文献 42](#_Toc1171030834_WPSOffice_Level1)

# 1.绪论

## **1.1** 研究背景

近十年多来，随着中国手机用户的数量爆炸增长，中国的手机用户基数已经十分庞大。根据中国工信部的数据显示，2018年中国手机用户总数已达15.7亿。如此庞大的用户群体，必然产生海量的通信数据。在保障用户隐私的前提下，人们很早就有分析这些数据的想法，而数据挖掘和机器学习技术的进步让人们有了更好的工具去从海量的数据中发现有价值的知识。比如建立电信客户流失模型，实现精确营销，分析客户关系等等都是数据挖掘在电信领域的成功应用。

数据挖掘[[[1]](#endnote-0)]（Data Mining）是从大量的、不完全的、有噪音的、模糊的、随机的实际应用数据中，提取隐含在其中人们事先不知道的、但又是潜在有用的信息和知识的过程。它是一个跨学科的计算机科学分支，与人工智能、机器学习、统计学和数据库等领域密不可分。

机器学习[[[2]](#endnote-1)]理论主要是设计和分析一些让计算机可以自动“学习”的算法。机器学习算法是一类从数据中自动分析获得规律，并利用规律对未知数据进行预测的算法。

本文将机器学习、数据挖掘技术应用到电信运营领域。实验的数据集为60万条已脱敏的中国移动公司的客户数据，其中有500条是被投诉的用户的信息。本文的目的是，利用机器学习的分类算法和异常检测模型，预测未来可能会被投诉的用户。

1.2 相关理论

1.2.1 分类算法

分类与预测是机器学习和数据挖掘的一个重要领域。分类算法是一种有监督的算法，通过对已知类别的数据进行训练，从中发现分类规则，预测新数据的类别。分类算法在金融、互联网等领域有着广泛的应用。常见的分类算法有基于贝叶斯公式的朴素贝叶斯、贝叶斯网络，基于神经网络的后向传播算法，和基于距离的kNN算法，基于决策树的id3，算法、c4.5算法，以及svm算法。

各个数据集的特点不同，有的数据集包含文本数据，有的只有数值型数据;有的数据集有几十万条数据，有的数据集只有几千条数据;有的数据集有几千个特征，有的数据集只有不到一百个特征。数据集的特点不同，选择的算法也不同。图1.2.1是机器学习库sklearn的官方给出关于选择分类算法的建议。

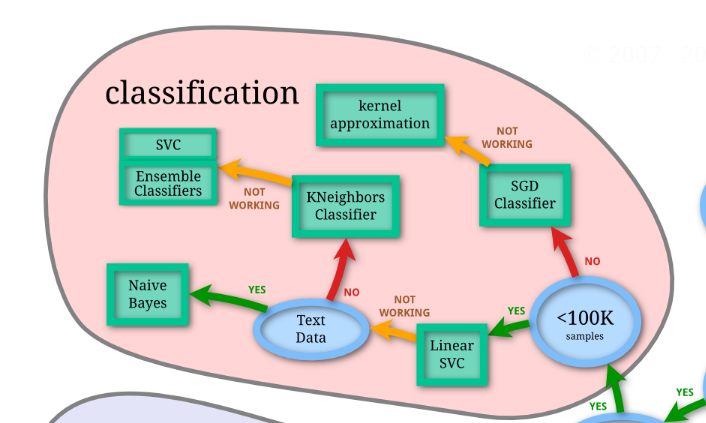


图1.2.1 分类算法选择

1.2.2 异常检测模型

在一些情况下，异常检测可以看成是不均衡数据的分类问题。但是异常检测模型与分类算法又有明显的不同。常见的异常检测模型大致有一下几类：

1. 统计假设检验

假设数据符合某个分布，该分布所定义的异常值即为异常数据。例如高斯异常检测：假定数据符合高斯分布，然后使用极大似然估计求出高斯分布的均值和方差;接着选择一个阈值，将待检测的数据代入高斯函数计算概率，如果概率小于该阈值，那么该数据就为异常数据。

这类方法虽然计算成本低，但是存在较强的假设，很多时候效果不一定很好。

1. 线性模型

假设数据可以映射到低维空间，那么定义在映射到低维空间后表现不好的数据为异常点。这类算法有用于异常检测的PCA和 one-class svm。

1. 相似度衡量模型

异常点与正常点相似度较低，通过计算相似度可以识别异常点。LOF、LOCI计算局部的数据密度来检测异常;Isolation Forest算法通过划分超平面来计算寻找异常点，在异常点所在的空间中，找到一个异常点需要的划分次数更少。ABOD通过计算每个样本与其他样本对所形成的夹角的方差，异常点的方差变化小。

## 1.3 数据轮廓

原始数据集中有600023位客户的数据，其中502位客户被投诉。数据有27维，其中6维是布尔型数据，名称分别为：'users\_3w', 'twolow\_users', 'roam\_users02', 'roam\_users01','vv\_type', 'in16\_roam\_tag'，其余21维为数值型数据，名称分别为'roam\_call\_duration','roam\_duration\_02','mon\_use\_days','is\_p\_app\_wx\_times', 'zhujiao\_time','zhujiao\_times','mb5','mb10','mb30','mb60','ma60','total\_count','beijiao\_times', 'use\_days', 'zhujiao', 'beijiao','zhujiao\_jt', 'open', 'close', 'open\_day', 'cell\_num'。

1.4 系统开发

1.4.1 硬件参数

系统运行平台的具体参数如下：

操作系统：Manjaro Linux x86\_64

CPU: Intel i7-8550U (8) @ 4.000GHz

RAM: 8G

1.4.2 开发技术

开发语言：python3.7

数据库：mongodb3.4

Web服务器：nginx

第三方库：

scikit\_learn==0.22.1,pyod=0.7.8,imbalanced\_learn==0.6.1,pandas==1.0.1,numpy==1.18.1,matplotlib==3.1.3,,seaborn0.10.0,pyqtgraph==0.10.0,flask1.1.1

1.4.3 系统架构

系统分为实验端和web端。

其中实验端用于对数据进行实验，使用分类算法和异常检测模型探索数据的内在结构。

实验端有7个模块：数据预处理、特征工程、数据可视化、模型评估、分类算法、异常检测模型。

Web端分为前端和后端。nginx作为前端服务器，后端使用python的flask框架进行编写。后端提供restful api，前端采用ajax和axios与后端交互。Web端共有5个模块：数据中心(上传数据、查看数据)，数据统计信息、数据可视化、训练、预测。Web端程序运行在docker容器之上，部署起来非常方便。

本项目分为实验端和web端两部分，实验主要使用的编程语言为Python 3.7.6

# 数据预处理

数据的来源有很多，网上电子银行的操作会产生数据，人们使用手机会产生数据，问卷调查也会产生数据，等等。但是这些数据往往是不干净、不完全、有噪音、不一致、冗余的。这些数据还不能被直接分析，我们需要对数据做很多预处理工作。数据预处理是数据分析的基础。

## 数据清洗

* + 1. 处理缺失数据

处理缺失数据是数据清洗的一项重要工作。数据缺失的原因有很多，可能是因为收集数据的机器损坏了，也可能是因为涉及数据提供者的隐私。

处理数据缺失的方法包括但不限于以下3种：

1. 忽视

把含有缺失数据的属性去除。但是如果缺失的数据太多，这种方式会导致可用的数据很少。

1. 手工填写

数据收集者可以重新收集缺少的数据，但是在一些情况下，人们往往无法重新收集缺失的数据，而且这种方法也会增加成本。

另外一种方式是根据领域知识去猜测缺失的数据。比如一个人的性别为男，年龄是19岁，那么可以推测出他的体重为60公斤左右。

但是手工填写缺失数据是一个枯燥繁琐的过程。

1. 自动填写

借助计算机的帮助自动填写缺失数据。比如在缺失的地方填个常数或者该属性的平均值、中位数、均值。

在该数据集中，各个属性的频次直方图类似图2.1.1。离群点很突出，采用中位数来替换缺失值。

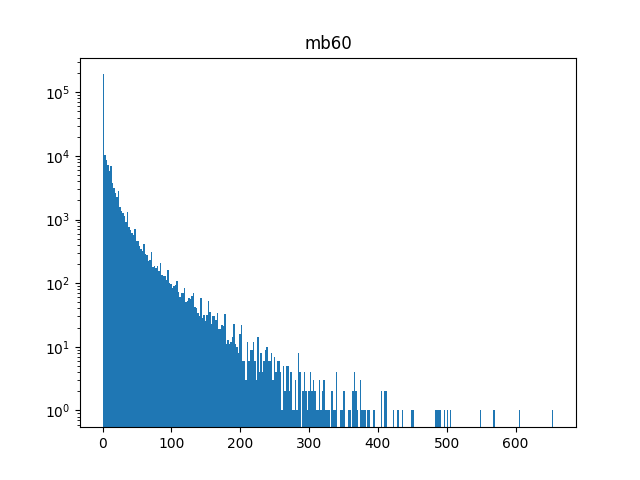


图2.1.1 mb60特征频次直方图

* + 1. 离群点检测

离群点是指不符合一般模型的数据对象，它们与数据的其它部分不同。如果不处理这些值，算法模型的结果可能会有很大的误差。

检测离群点的方法大致可分为一下3类：

1. 基于统计的离群点探测

假设数据符合某一个分布或者概率模型，然后根据模型采用不一致性检验：验证一个对象O关于分布F是否有显著不同。例如正态分布的3σ检测。

1. 基于距离的离群点探测

不依赖统计检验，基于距离的离群点探测将离群点看作是没有足够邻居的对象。如果一个数据点的近邻点小于一个给点的值，那么认为此数据点为异常点。

例如局部离群因子检测方法 (Local Outlier Factor, LOF)就是一个常见的基于距离的离群点探测算法。

1. 基于偏离的离群点探测

这类方法通过检查一组对象的主要特征来确定孤立点 ，如果与给出的描述偏离大的对象被认为是孤立点 。

观察本数据集某特征的分位数图（图2.1.2-1），我们发现，数据存在明显的离群点，而经过检查，这些离群点代表的用户并非被投诉的用户，所以我们用中位数代替这些离群点。处理后的数据的分位数图如2.1.2-2所示，不存在离群点。

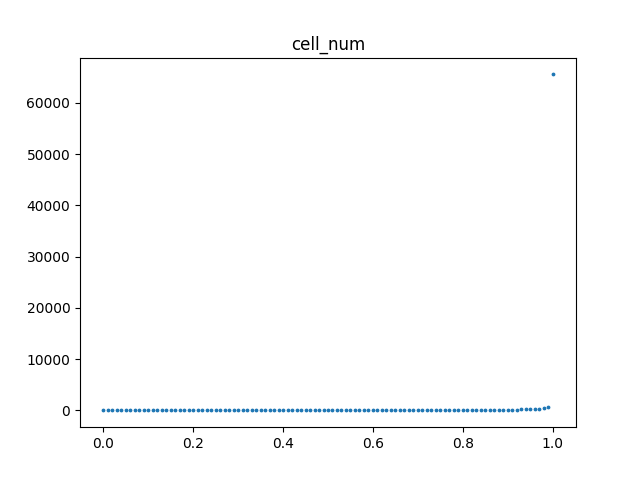


图2.1.2-1 cell\_num特征分位数图

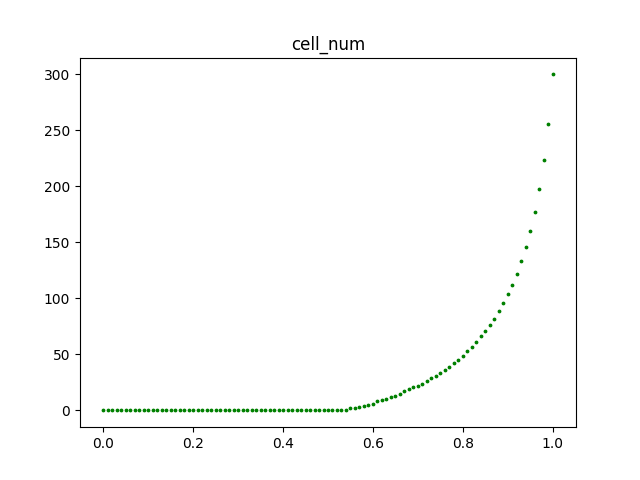


图2.1.2-2 cell\_num特征分位数图（去离群点）

* 1. 数据无量纲化

数据无量纲化是指将不同规格的数据转换到同一规格，或者将不同分布的数据转换到某个特定分布。无量纲化不仅可以加快模型运算速度，还可以提升模型的精度。数据无量纲化的方式大概有以下两种。

1. 数据规一化(Normalization):

数据规一化对原始数据进行线性变换。假定和分别为特征A的最小值和最大值。数据归一化通过计算

（式2.2-1）

将A的值v映射到区间[] 中的。

1. 数据标准化(Standardization,又称Z-score normalization)

数据标准化将属性 A 的值基于 A 的平均值和标准差规范化。假定和分别为特征A的平均值和标准差，则计算

（式2.2-2）

将A的值v转换为。

# 特征工程

特征工程是利用领域知识，通过数据挖掘技术从原始数据中提取特征的过程。这些特征可以用来提高机器学习算法的性能。

特征工程包括一下三大部分：

1. 特征提取(feature extraction)

特征提取是指从文字数据、声音数据等非结构化的数据中提取新的信息作为特征。比如从从一张图中提取出它的颜色特征、纹理特征和代数特征。

1. 特征创造(feature creation)

特征创造是把现有特征进行组合，或互相计算，得到新的特征。比如将人的体重特征(千克)除以身高特征(平方米)，就得到了人的体质指数(Body Mass Index)。

1. 特征选择(feature selection)

特征选择是指从所有特征中选择出有意义，对模型有帮助的特征的过程。比如说，如果一个特征中的大多数值都一样，或者这个特征的方差很少，那我们就可以过滤掉这个特征。

* 1. 特征选择
     1. 方差过滤

如果一个特征的方差很小，那么样本在这个特征上的差异也很小。选定一个阈值，方差过滤将方差低于该阈值的特征踢除。过滤法的主要目的是:在维持算法表现的同时,通过减少特征维度来降低算法的计算成本。

对于本数据的特征集，计算各个特征的方差如表3.1.1。

表3.1.1 特征方差

|  |  |
| --- | --- |
| 特征 | 方差 |
| users\_3w | 0.365218 |
| twolow\_users | 0.250742 |
| roam\_users02 | 0.129258 |
| roam\_users01 | 0.189707 |
| vv\_type | 0.340842 |
| in16\_roam\_tag | 0.265152 |
| roam\_call\_duration | 42.130754 |
| roam\_duration\_02 | 3273.371141 |
| mon\_use\_days | 13.447850 |
| is\_p\_app\_wx\_times | 1501.844455 |
| zhujiao\_time | 7723.583749 |
| zhujiao\_times | 88.860545 |
| mb5 | 3.474165 |
| mb10 | 12.031849 |
| mb30 | 38.792758 |

续表3.1.1

|  |  |
| --- | --- |
| 特征 | 方差 |
| mb60 | 19.882516 |
| ma60 | 25.651550 |
| total\_count | 154.629745 |
| beijiao\_times | 78.777257 |
| use\_days | 11.535426 |
| zhujiao | 147.391548 |
| beijiao | 2.140705 |
| zhujiao\_jt | 221.272752 |
| open | 6.160473 |
| close | 6.904983 |
| open\_day | 13.843749 |
| cell\_num | 55.570052 |

观察上表发现，对于前5行布尔型特征，他们的方差非常小。事实上，如图3.1.1所示，各个布尔型特征中，0值出现的频率都在80%以上。所以我们可以推定，这5个布尔型特征对于模型没有帮助，我们可以剔除掉这5个特征。

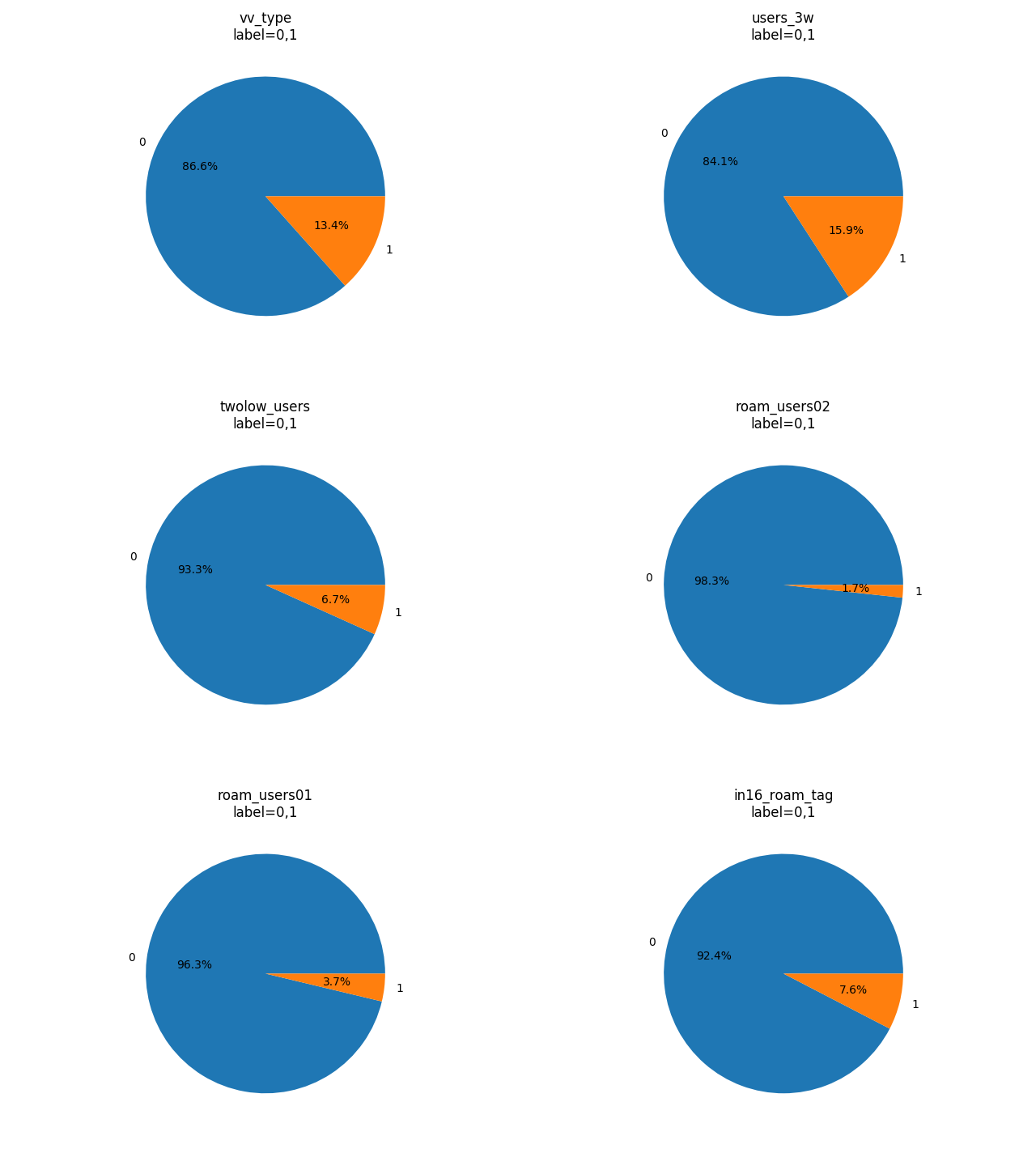


图3.1.1 各布尔型特征0,1比重

* + 1. 卡方检验

卡方检验的目的是检验定性自变量对定性因变量的相关性。卡方检验的无效假设是：观察频数与期望频数与期望频数没有差别。令A代表某个类别的观察频数，E代表基于计算出的期望频数，A与E之差称为残差。则

（式3.1.2）

这个统计量就表示自变量对因变量的相关性。

sklearn中的feature\_selection模块提供了进行卡方检验的方法chi2,该方法返回卡方值和P值两个统计量,因为我们很难界定卡方值的有效范围,所以我们通过p值来判断自变量对因变量的相关性。我们一般使用0.01或0.05作为显著性水平。即只有当p或0.01时，我们才认为自变量和因变量有相关性。

对于本数据集，计算结果如表3.1.2，可以发现，所有的p值都小于0.01。

表3.1.2 卡方检验

|  |  |
| --- | --- |
| 特征 | P值 |
| roam\_call\_duration | 0 |
| roam\_duration\_02 | 0 |
| mon\_use\_days | 0 |
| is\_p\_app\_wx\_times | 0 |
| zhujiao\_time | 0 |
| zhujiao\_times | 0 |
| mb5 | 0 |
| mb10 | 0 |
| mb30 | 0 |
| mb60 | 0 |
| ma60 | 0 |
| total\_count | 0 |
| beijiao\_times | 0 |
| use\_days | 0 |
| zhujiao | 0 |
| beijiao | 0 |

续表 3.1.2

|  |  |
| --- | --- |
| 特征 | P值 |
| zhujiao\_jt | 0 |
| open | 0 |
| close | 0 |
| open\_day | 0.001 |
| cell\_num | 0 |

* + 1. 互信息法

互信息法的目的是捕捉每个特征与标签之间的任意关系(包括线性和非线性关系)。定义两个随机变量X,Y, 若P(X,Y) = P(X)P(Y) 则随机变量X，Y独立。随机变量的独立性反应了它们之间是否有关系，但不能刻画关系的大小。定义互信息为：

（式3.1.3）

互信息既可以反应两个随机变量之间是否有关系，也能反应它们之间关系的强弱。

sklearn的feature\_selection.mutual\_info\_classif提供了实现互信息分类的方法，它的返回每个特征与标签之间的互信息量估计，这个估计量在[0,1]区间之间，值为0则表示两个变量独立,为1则表示两个变量完全相关。

对于本数据集，计算结果如表3.1.3所示。可以发现，所有特征与标签之间的互信息量都比较小。

表3.1.3 互信息法

|  |  |
| --- | --- |
| 特征 | 互信息量 |
| roam\_call\_duration | 0.001 |
| roam\_duration\_02 | 0 |

续表 3.1.3

|  |  |
| --- | --- |
| 特征 | 互信息量 |
| mon\_use\_days | 0.006 |
| is\_p\_app\_wx\_times | 0.000 |
| zhujiao\_time | 0.001 |
| zhujiao\_times | 0.002 |
| mb5 | 0.002 |
| mb10 | 0.002 |
| mb30 | 0.002 |
| mb60 | 0.002 |
| ma60 | 0.001 |
| total\_count | 0.001 |
| beijiao\_times | 0.000 |
| use\_days | 0.002 |
| zhujiao | 0.002 |
| beijiao | 0 |
| zhujiao\_jt | 0.001 |
| open | 0.001 |
| close | 0.001 |
| open\_day | 0.006 |
| cell\_num | 0.001 |

* 1. 降维

降维算法是指采用某种映射方法，在保留大部分信息的前提下，将原高维空间的数据映射到低维空间中。令R表示m行n列的数据矩阵，降维的本质是学习一个映射函数， (，将原始的数据从n个维度降低到d(个维度。降维算法可以减少信息冗余造成的误差，提高算法精度，也可以减少算法计算成本。

常见的机器学习降维算法有：主成分分析算法(PCA)，线性判别分析(LDA)，局部线性嵌入(LLE),和拉普拉斯特征映射(Laplacian Eigenmaps)。针对本数据集，我们采用主成分分析和线性判别分析对数据降维。

* + 1. 主成分分析

主成分分析算法通过线性投影将高维的数据映射到低维的空间中表示，并使降维后的数据的方差最大。这样在保留住较多原始数据的信息的同时，减少了数据的维度。。

在3.1.1节已经讲过，对于本数据集，只有数值型特征对模型有帮助。所以我们采用PCA算法，对21维的数值型特征进行降维。结果发现，在保留99%的原始数据的信息下，数据可以降低到3维。我们使用python的pyqtgraph库做3d图(其中红色的数据点表示被投诉的用户)，如下：

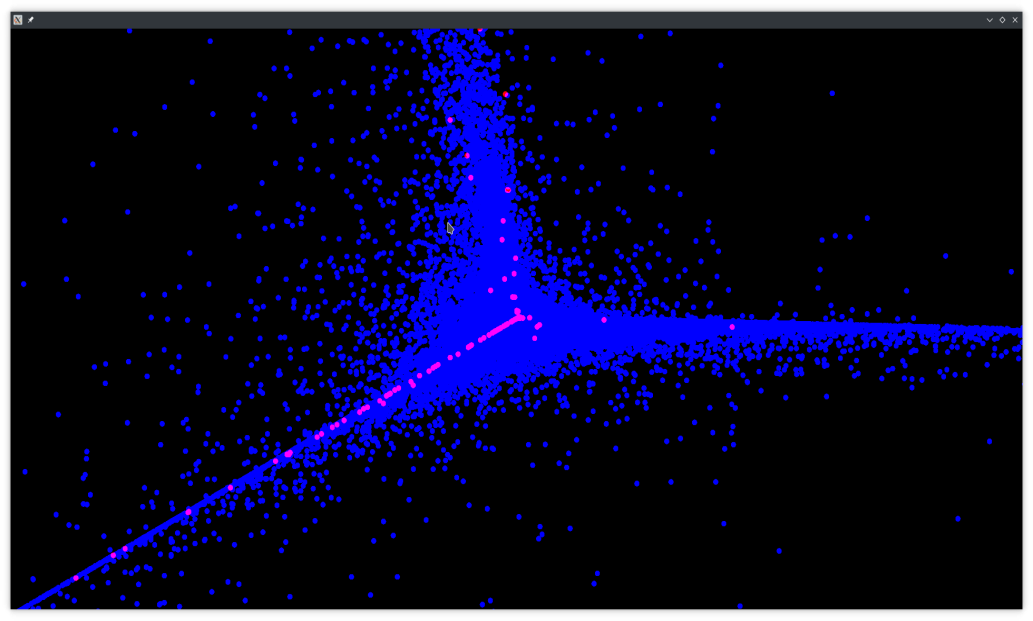


图3.2.2-1 3维数据立方图

使用python的matplotlib库做3个维度之间的散点图如下((其中红色的数据点表示被投诉的用户)：

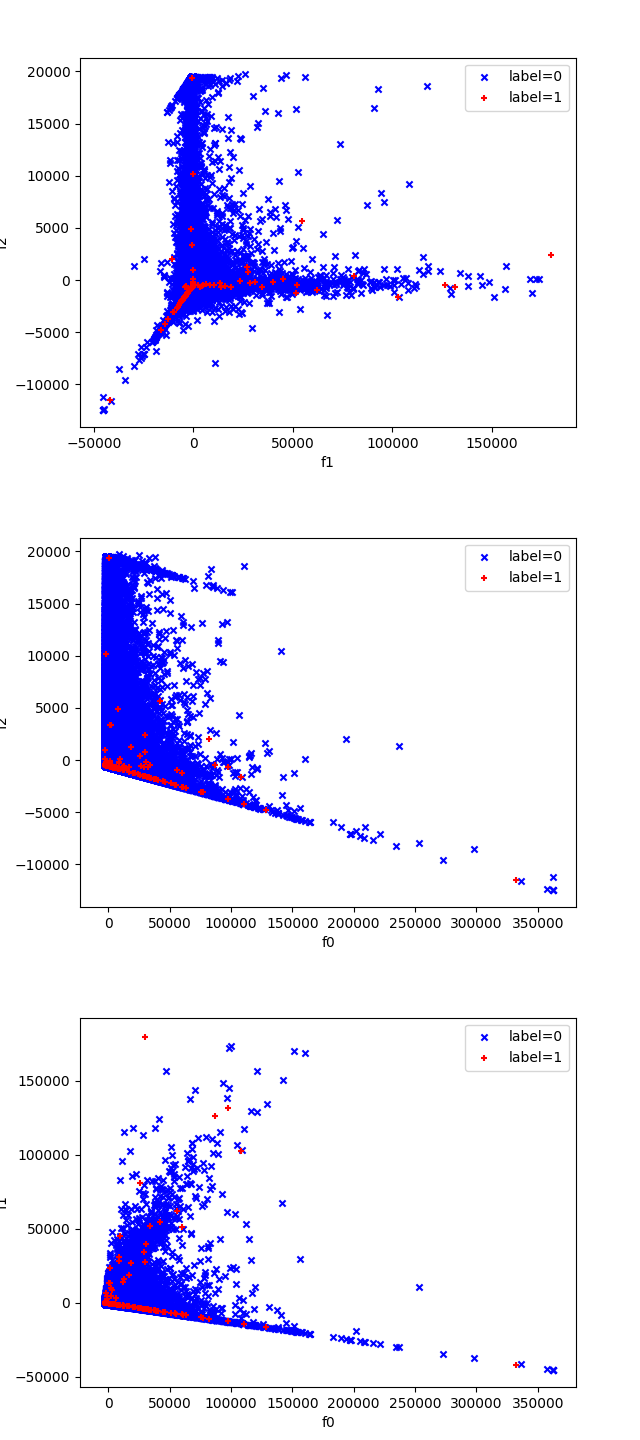


图3.2.2-3 三个维度之间的散点图

* + 1. 线性判别分析

PCA算法的缺点在于它无法探究数据内在的结构，对于一些分类数据无法很好地处理。例如图3.2.3-1这样的二分类数据，如果使用PCA算法降维，则两类数据会揉杂在一起，变得无法区分。

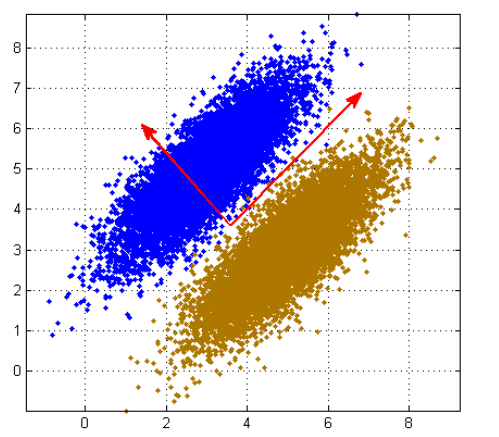


图3.2.3-1 二分类数据

线性判别分析(Linear Discriminant Analysis)是一种有监督的线性降维算法，它的目标不是保持原始数据最多的信息，而是使得降维后的数据更容易被区分。LDA算法的基本思想是投影，它通过找出特征向量w,将k类m维数据投影到另一个更低维的空间，使得投影后各个类之间的距离尽可能远，同类数据的距离尽可能小。

假设数据为二分类数据(正类和负类)，令和分别表示正类数据和负类数据的个数， 和分别表示正类和负类的均值，表示第i个正类数据( i=1,2,3,...,)，表示第i个负类数据( i=1,2,3,...,)。则求最优特征向量 w 的问题就转换成求下式的解：

（式3.2.2-1）

其中，

（式3.2.2-2）

(式3.2.2-2）

对于本数据集，使用sklearn库提供的LinearDiscriminantAnalysis类进行线性判别分析，

结果如图3.2.3-2所示。其中红色表示正类数据，蓝色表示负类数据。可以发现，数据的类别特征并不明显。

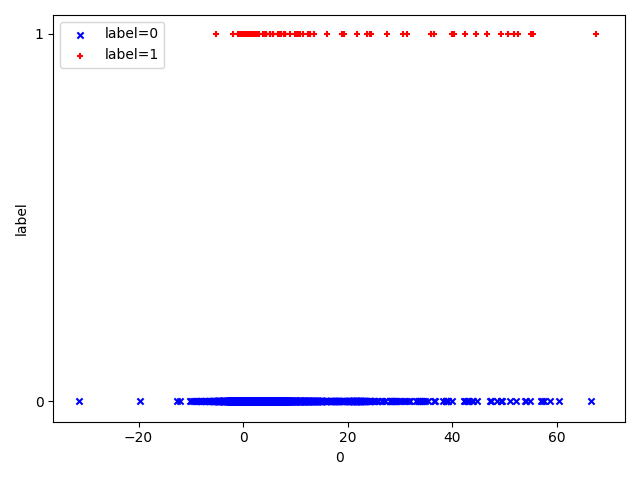


图3.2.3-2 线性判别分析结果可视化

# 模型选择与评估

* 1. 交叉验证

在同样的数据集上训练和测试是一种错误的做法，这会导致过拟合(over fitting)的问题：训练出的模型在已有的数据集上表现得很好，但是对于尚未出现的数据，模型的表现则非常差。

为了解决过拟合的问题，一个解决方法是将数据分为训练集(60%)和测试集(40%)。模型采用不同的参数在训练集上训练，在测试集上进行测试参数，选择表现最优的参数。但是这种方法在测试集上可能会出现过拟合的问题。避免这个缺点的方法是，准备一部分验证集数据。模型训练完后在验证集上调参，最后在测试集上对模型进行最后的评估。

然而，将数据分为3个数据集合会大大减少模型用来训练的数据。为了解决这个问题，sklearn提供了一种k-折叠交叉验证的方法。过程如下：

1. 将原始数据平均分为k份。
2. 选一份数据做测试集来评估模型的表现，其余部分做训练集。
3. 重复第(2)步直到k份数据都被选中过做测试集。
4. 计算每一步模型表现的平均值。
   1. 模型评估

对于不平衡的数据，采用准确率做为模型评估的方法是做错误的。例如在某个二分类问题中，有99%的数据是正类，如果模型把所有的数据都预测为正类，那么它的准确率将会为0.99，然而该模型对于这个问题并没有提供任何有价值的解决方案。一般而言，对于不均衡的数据，我们有以下三种模型评估的方式：

1. 混淆矩阵
2. ROC曲线
3. AUC面积
   * 1. 混淆矩阵

对于二分类问题，模型需要预测样本的是负类(negative)还是正类(positive)，预测可能是正确的，也可能是错误的。这样就得到了四个基础指标。

1. True Positive (TP)：模型预测为正类(positive)，预测正确，样本实际为正类(positive)。
2. False Negative (FN)：模型预测为负类(negative), 预测错误，样本实际为正类(positive)。
3. False Positive (FP)：模型预测为正类(positive)，预测错误，样本实际为负类(negative)。
4. True Negative (TN): 模型预测为负类(negative)，预测正确，样本实际为负类(negative)。

以上四个基础指标组合在一起就是混淆矩阵，如表4.2.1所示。

表4.2.1 混淆矩阵



在以上指标上，我们定义另外3个新的指标：

1. 精确率 (Precision):在模型预测为positive的所有结果中，模型预测对的比重。

（式4.2.1-1）

1. 召回率(Recall),或者称灵敏度(Sensitivity):在所有正类样本中，模型预测对的比重。

（式4.2.1-2）

1. 特异度(Specificity):在所有负类样本中，模型预测对的比重。

（式4.2.1-3）

一般而言，Precision和Recall的值越高，说明模型的表现越好。为了综合Precision和Recall，我们提出一个新的指标，叫做F1 Score。

（式4.2.1-4）

F1 Score的取值范围在[0,1]区间，数值越大说明模型表现越好。

* + 1. ROC曲线与AUC面积

ROC曲线指受试者工作特征曲线 （receiver operating characteristic curve）。对于每个数据x，模型会输出该数据属于正类的概率为P(1|x)。我们设定一个阈值T（1），只有当P(1|x)T时，模型才将该数据归为正类。

令FPR = FP/(FP + TN)，即负样本中的错判率;TPR = TP/(TP + TN)，即判对样本中的正样本率。

那么每个T（1）值都会对应一个FPR值和TPR值。当T从0变化到1时，我们得到一条横坐标为FPR，纵坐标为TPR的曲线，这条曲线就是ROC曲线(如图4.2.2)。

一般而言，如果模型输出的TPR值越高而FPR值越低，则说明这个模型表现越好。我们定义AUC值为ROC曲线下的面积。AUC的值一般大于0.5,越接近1说明模型的表现越好。

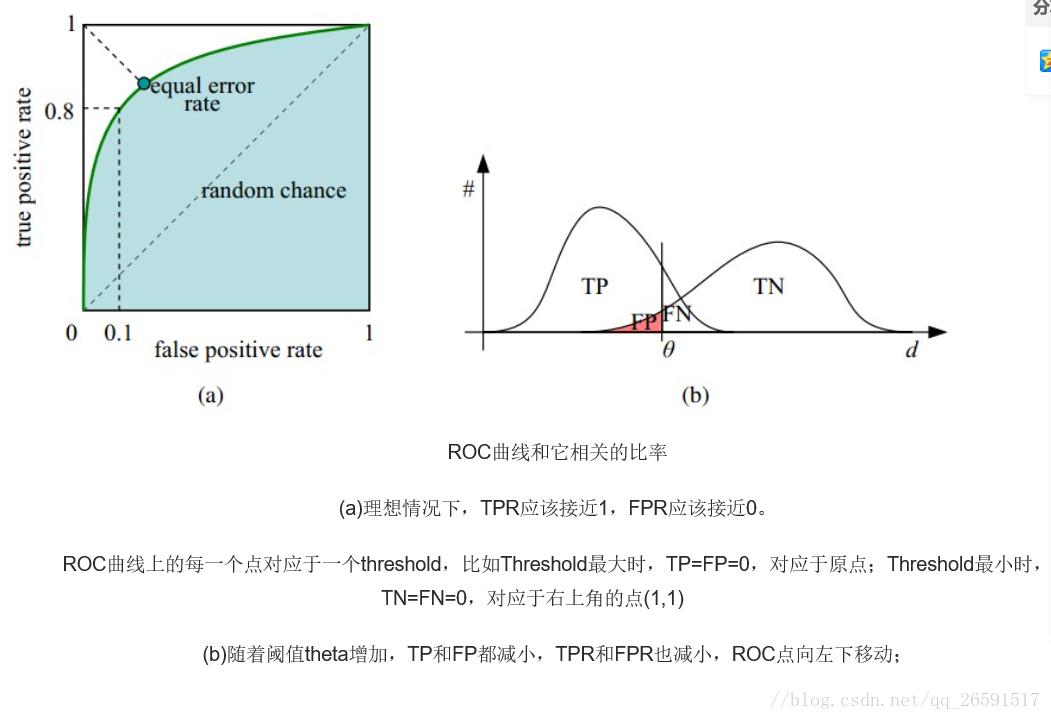


图4.2.2 ROC曲线示例

# 不均衡数据处理

一般而言，机器学习和数据挖掘采用的分类算法都假设数据不同类的比例是均衡的。然而显示生活中我们经常会遇到不平衡的数据集，比如信用卡欺诈数据，广告点击数据，等等。对于不平衡的数据，尤其是本案例中这样极度不平衡的数据（正类与负类的比约为7：10000），如果没有处理好，模型输出的效果会非常差。

处理不平衡数据，一般可以从以下两方面考虑：

1. 改变数据分布，让数据变得均衡。
2. 代价敏感学习，不同类别的的误分类代价不同，让模型更注重少数类。
   1. 数据采样
      1. 随机下采样

让不均衡的数据变得均衡的最简单的方法是随机减少多类样本。这样的方法操作简单，计算成本低，但是会浪费大量数据，使得数据集极大程度缩小。

* + 1. Tomek Link方法

对于不同类别的样本x和y，如果它们满足以下条件，那么它们叫做TomelLinks：

不存在另外一个样本z, 使得d(x,z) < d(x,y) 或者 d(y,z) < d(x,y)成立，其中d(a,b)表示两个样本之间的距离。

TomekLinks之间互为近临关系，但是它们又属于不同的类别，这说明它们很有可能是噪声数据, 或者处于两类的边界上。

对于TomekLinks，我们可以移除属于多数类的那个样本，以此让数据变得均衡。

* + 1. NearMiss方法[[[3]](#endnote-2)]

NearMiss方法是利用距离剔除多数类的方法，具体有以下3类：

1. NearMiss-1:在多数类样本中选择与最近的3个少数类样本的平均距离最小的样本
2. NearMiss-2：在多数类样本中选择与最远的3个少数类样本的平均距离最小的样本。
3. NearMiss-3：对于每个少数类样本，选择离它最近的给定数量的多数类样本。

这三类算法各有优缺点，类别比例不同，算法的效果也不同。一般而言，NearMiss-2的效果可能更好。

* + 1. 随机上采样

让不均衡的数据变得均衡的另外一种简单的方法是随机增加少类样本。与随机下采样类似，随机上采样方法操作简单，计算成本低。但是随机上采样会使得重复样本过多，容易造成分类器过拟合。

* + 1. SMOTE算法[[[4]](#endnote-3)]

SMOTE(Synthetic Minority Over-sampling Technique)，又称合成少数类过采样技术，它的主要思想是通过在一些位置相近的少数类样本中生成新的样本。SMOTE算法的步骤如下：

1. 对于每个少数样本x，计算它到其它少数类样本的距离，得到k近邻。
2. 对于每个少数类样本，从k近邻中随机选择若干样本。
3. 对于每一个随机选出的近邻，按如下公式构建新的样本：

（式5.1.5）

其中为区间(0,1)内的一个随机数。

SMOTE的一个缺点在于它容易产生边缘模糊化的问题。如果一个少数类样本处于该类的分布边缘，那么由此样本和它相邻样本产生的新样本也会处于边缘，并且越来越边缘化，从而导致多数类和少数类样本的边界变得模糊化。

* + 1. 综合采样

综合采样是指将上采样和下采样结合，解决类别分布不平衡的问题。主要有两种方法：SMOTE+TomekLinks和SMOTE+ENN。

1. SMOTE+TomekLinks

SMOTE+TomekLinks方法首先利用SMOTE方法产生新的少数类样本，生成新的数据集。然后剔除新数据集的TomekLinks对。

1. SMOTE+ENN

SMOTE+ENN方法首先利用SMOTE方法产生新的少数类样本，生成新的数据集。然后对于新数据集中的每一个样本使用kNN(一般k取3)方法预测，若预测结果与实际不同，则从新数据集中剔除该样本。

综合采样可以克服SMOTE算法的一些缺点，使得模型的精度更高，但是也极大地增加了计算成本。

* + 1. 总结

Python的imbalanced-learn库提供了大量处理不平衡数据的方法。利用它，我们可以很容易地进行不均衡数据处理。由于下采样算法会浪费大量数据，所以对于上述算法，我们只使用随机上采样、SMOTE、SMOTE+TomekLinks和SMOTE+ENN进行不均衡数据处理的试验，然后使采用sklearn 中的linear\_model模块的SGDClassifier类对数据进行分类，观察效果。

试验结果如表5.1.7。

表5.1.7 不均衡处理算法总结

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算法 | 时间 | precision | recall | f1 score | auc |
| 随机上采样 | 0.51s | 0.0007 | 0.92 | 0.002 | 0.79 |
| SMOTE | 0.89s | 0.002 | 0.86 | 0.004 | 0.81 |
| SMOTE+TomekLinks | 5021.20s | 0.003 | 0.84 | 0.005 | 0.82 |
| SMOTE+ENN | 4849.65s | 0.002 | 0.85 | 0.005 | 0.80 |
| 不采样 | —— | 0.006 | 0.45 | 0.013 | 0.81 |

* 1. 代价敏感学习

代价敏感学习从算法层面解决不平衡数据带来的问题。代价敏感学习的核心是代价矩阵。在实际应用中，不同类型的分类错误的代价是不同的，比如一般情况下，“将癌症病人误疹为健康人”比“将健康人误疹为癌症病人”的代价更高。定义代价矩阵如表5.2，其中表示为将类别j误分为类别i的代价。

表5.2 代价矩阵

|  |  |  |
| --- | --- | --- |
| Predict  True | 0 | 1 |
| 0 | 0 |  |
| 1 |  | 0 |

利用代价矩阵，我们得到优化目标为：

（式5.2）

其中x是一个样本，P(j |X）表示在算法中获得的 x 属于类别 j 的后验概率。这就使得模型不仅要考虑x为j类的概率，还要考虑将类别j误分为类别i的代价。

# 实验结果

* 1. 分类模型

使用sklearn提供的RandomForest、LinearSVC、SGDClassifier和DecisionTreeClassifier(CART)算法对数据进行分类，模型评估采用precision，recall，f1 score和auc值。最终的实验结果如表6.1.5。

* + 1. 决策树

ID3,C4.5,和CART是3种主流的决策树算法。sklearn的tree模块的DecisionTreeClassifier提供了CART算法的实现。

经过实验（结果图6.1.1-2）发现，如果不对数据进行不均衡处理，模型输出的f1 score值可以达到0.80,而一旦对数据进行不均衡处理，模型的输出效果立马变得很差。这说明CART算法可以很好地处理不均衡数据，同时也说明对数据进行不均衡处理并不总能提高模型的精度。

在不对数据进行不均衡处理的情况下，算法的精度和是否对数据降维没有关系。但是如图6.1.1-3所示,降维的确能加快算法的运行速度。

由于模型输出的决策树非常大，这里只展示输出决策树的局部，如图6.1.1-1。

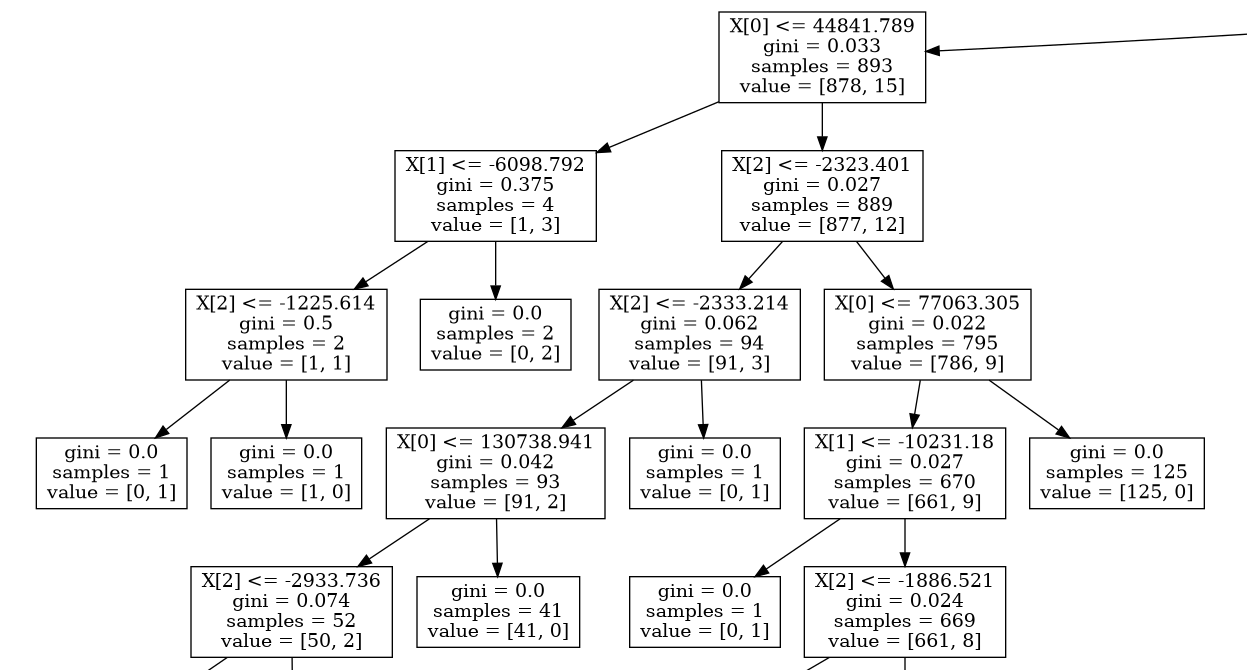


图6.1.1-1 CART输出决策树的局部图

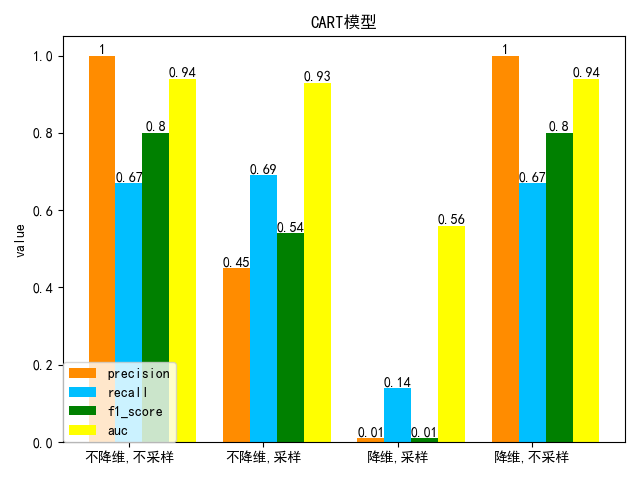
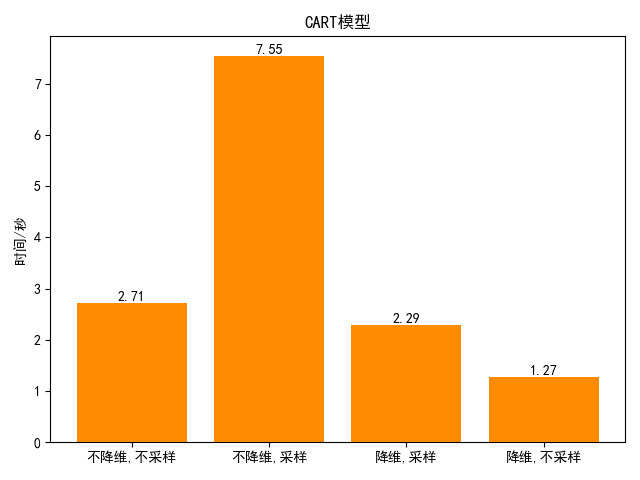


图6.1.1-2 CART模型评估



6.1.1-3 CART模型运行效率

6.1.2 随机森林

随机森林是一种集成学习的方法，随机森林由多棵决策树组成，大多数决策树输出的结果就是随机森林输出的结果。sklearn的RandomForestClassifier类提供了该算法的实现。

实验结果如图6.1.2-1和6.1.2-2所示。我们发现相比决策树算法，随机森林的精度并没有得到很好的提升，但是运行时间却大大增加了。

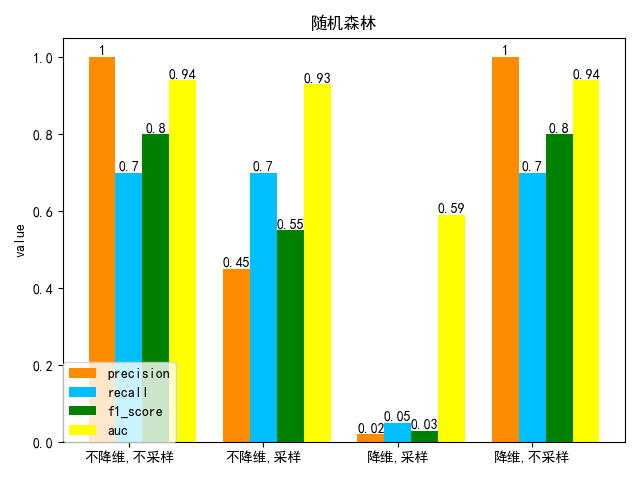


图6.1.2-1 随机森林模型评估

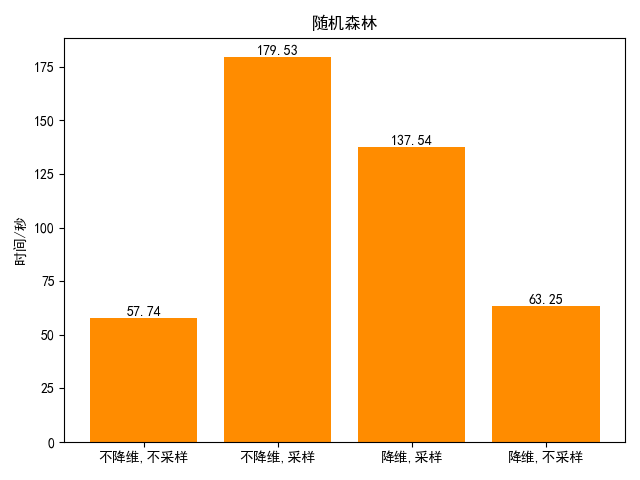


图6.1.2-2 随机森林运行效率

6.1.3随机梯度下降

由于样本数量十分庞大，如果采用普通的梯度下降算法，消耗的时间会非常多。而随机梯度算法每次从样本抽取一组进行梯度下降，这样可以大大减少算法运算的时间。

实验结果如表6.1.2所示，可以发现该模型对于这类数据的输出效果很差。

表6.1.2 随机梯度下降

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 算法 | 降维 | 不均衡处理 | precision | recall | f1 score | auc | 时间(秒) |
| SGDClassifier | False | False | 0.001 | 0.81 | 0.002 | 0.66 | 4.90 |
| SGDClassifier | False | True | 0.001 | 0.72 | 0.02 | 0.71 | 25.53 |

续表6.1.2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 算法 | 降维 | 不均衡处理 | precision | recall | f1 score | auc | 时间(秒) |
| SGDClassifier | True | True | 0.001 | 1.0 | 0.001 | 0.50 | 38.92 |
| SGDClassifier | True | False | 0.001 | 0.71 | 0.001 | 0.47 | 1.29 |

6.1.4 支持向量机

支持向量机算法是一机器学习中一类经典的算法。在skrearn中内置了4种核函数：线性核函数、多项式核函数、高斯核函数、Sigmoid核函数。由于数据集庞大，这里只使用线性核函数。

由表6.1.5第一行可以发现，面对30万行的训练集，即使是使用线性核函数，SVM算法运算的时间也会达到大约70分钟，而且算法的精度并不理想。所以对于该数据集，SVM算法并不是一个理想的选择。

6.1.5 总结

在尝试了以上模型后，我们将结果总结如表6.1.5。为了对各个模型的精度有更直观的感受，选择模型的最优表现数据（优先考虑f1\_score值，然后考虑时间，最后参考auc值），作图6.1.5-1。同时对比各个模型运算时间，如图6.1.5-2。

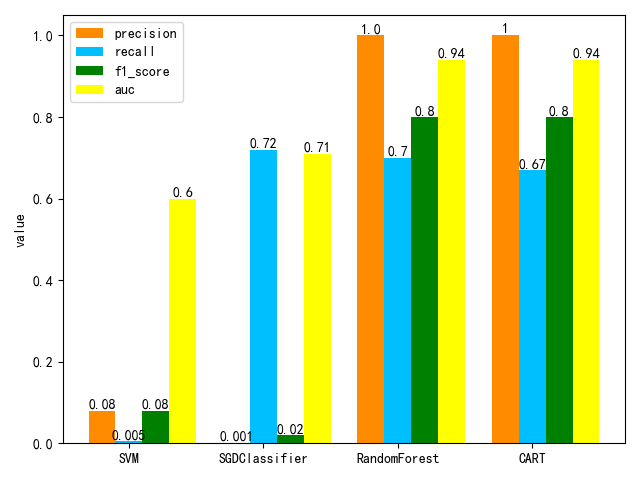


图6.1.5-1 分类模型精度对比

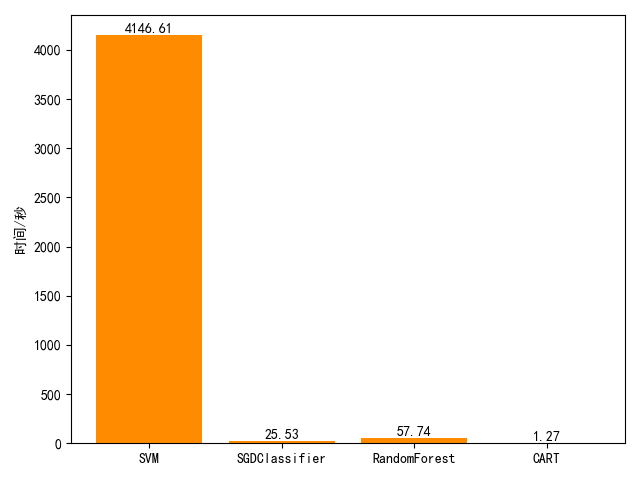


图6.1.5-2 分类模型时间对比

表6.1.5 分类模型总结

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 算法 | 降维 | 不均衡处理 | precision | recall | f1 score | auc | 时间(秒) |
| SVM  (linear kernel) | True | False | 0.08 | 0.005 | 0.008 | 0.60 | 4146.61 |
| SGDClassifier | False | False | 0.001 | 0.81 | 0.002 | 0.66 | 4.90 |
| SGDClassifier | False | True | 0.001 | 0.72 | 0.02 | 0.71 | 25.53 |

续表6.1.5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 算法 | 降维 | 不均衡处理 | precision | recall | f1 score | auc | 时间(秒) |
| SGDClassifier | True | True | 0.001 | 1.0 | 0.001 | 0.50 | 38.92 |
| SGDClassifier | True | False | 0.001 | 0.71 | 0.001 | 0.47 | 1.29 |
| RandomForest | False | False | 1.0 | 0.70 | 0.80 | 0.94 | 57.74 |
| RandomForest | False | True | 0.45 | 0.70 | 0.55 | 0.93 | 179.53 |
| RandomForest | True | True | 0.02 | 0.05 | 0.03 | 0.59 | 137.54 |
| RandomForest | True | False | 1 | 0.70 | 0.80 | 0.94 | 63.25 |
| CART | False | False | 1 | 0.67 | 0.80 | 0.94 | 2.71 |
| CART | False | True | 0.45 | 0.69 | 0.54 | 0.93 | 7.55 |
| CART | True | True | 0.01 | 0.14 | 0.01 | 0.56 | 2.29 |
| CART | True | False | 1 | 0.67 | 0.80 | 0.94 | 1.27 |

* 1. 异常检测模型

使用高斯异常检测算法、孤立森林算法和局部异常因子算法对数据做异常检测。实验结果如表6.2。通过图6.2-1和图6.2-2可以直观感受到各个模型的差异。

表6.2 异常检测模型总结

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算法 | precision | recall | f1 score | auc | 时间 |
| 孤立森林 | 0.01 | 0.82 | 0.02 | 0.94 | 3.95 |
| 高斯异常检测 | 0.001 | 0.001 | 0.001 | 0.80 | 7.11 |
| 局部异常因子 | 0.003 | 0.34 | 0.005 | 0.62 | 1936.57 |

# 

图6.2-1 异常检测模型精度对比

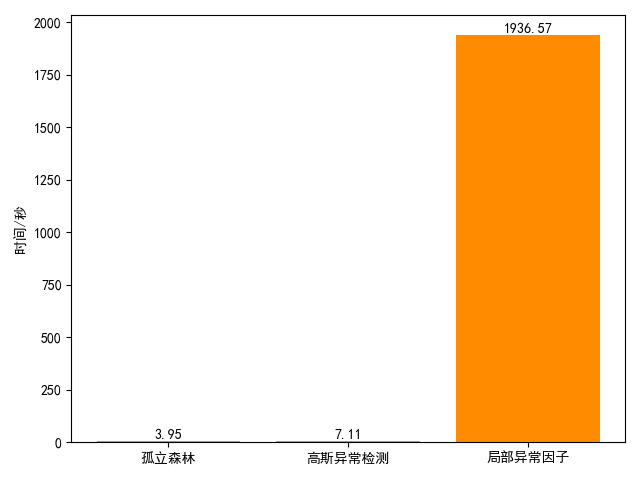


图6.2-2 异常检测模型时间对比

# 总结

为期半年的毕业设计已经结束，本科四年的学习生活也走向了尾声。

虽然我以前接触过数据挖掘的课，但我从来没有做过数据科学相关的实践。这是我数据科学的入门实践。为了完成这个系统，我首先在网上完成了斯坦福大学吴恩达教授在coursera发布的机器学习入门课程，然后在网上阅读很多其它数据科学方面的资料。

部署模型也是一件很耗费体力的事情，为了能够将系统发布到互联网上，在已经有的web开发的基础知识上，我还学习了mongodb、docker、flask等方面的知识。

过程是辛苦的，收获是巨大的。通过这次实践，我不仅学习到很多数据科学方面上网知识，还锻炼了自己解决问题的能力。我向我自己以及所有帮助过我的老师同学表示感谢。

# 致谢

感谢我的父母，这篇论文完成与covid-19病毒爆发期间，父母对我生活上的照顾让我能够心无旁骛地研究问题，父母对我精神上的鼓励使得我能够克服各种困难。在此我由衷地向他们表示感谢。

感谢我的指导老师，本论文的完成离不开老师对我独立解决问题能力的培养。在我遇到思维上瓶颈时是他的教诲让我茅塞顿开，在我懈怠时是他的鞭策让我继续前行。

感谢我的同学与网络上的朋友，他们对本论文的完成提供了宝贵的意见，解决了很多让我感到困惑的问题。

参考文献：

1. 环球网.工信部：2018年中国手机用户总数达15.7亿[EB/OL].https://m.huanqiu.com/article/9CaKrnKjkWT,2019-3-27.
2. Pang-Ning Tan , Michael Steinbach,Vipin Kumar.数据挖掘导论[M].人民邮电出版社:北京,: 2010-12-10:51.
3. Jiawei Han;Micheline Kamber.数据挖掘概念与技术[M].机械工业:北京,2006-4:1.
4. 周志华.机器学习[M].清华大学出版社:北京,2016-1-1:1.
5. 李航.统计学习方法[M].清华大学出版社:北京,2012-3:1.
6. 魏贞原.机器学习——Python实践[M].电子工业出版社:北京,2018-1:1.
7. Miguel Grinberg.Flask Web开发：基于Python的Web应用开发实战[M].人民邮电出版社:北京,2014-12:1.
8. 杨保华,戴王剑,曹亚仑.Docker 技术入门与实战[M].机械工业出版社:北京,2015-1:1.
9. Michael Dirolf,Kristina Chodorow.MongoDB权威指南[M].人民邮电出版社:北京,2011-5:1.
10. Divya Susarla,Sinan Ozdemir.特征工程入门与实践[M].人民邮电出版社:北京,2019-5-27:1.
11. 周敏奇,王晓玲,金澈清,等.Hadoop权威指南（第2版）[M].清华大学出版社:北京,2011-7:100.
12. C. S. 霍斯曼.快学Scala[M].电子工业出版社:北京,2012-10:1.
13. Julie Steele,Noah Iliinsky.数据可视化之美[M].机械工业出版社:北京,2011-6:1.
14. Jake VanderPlas.Python数据科学手册[M].人民邮电出版社:北京,2018-1-20:1.
15. Eli Bressert.SciPy and NumPy[M].O'Reilly Media:加利福尼亚,2012-11-22:1.

[16]Wes McKinney.Python for Data Analysis[M].O'Reilly Media:加利福尼亚,2012-11-1:1.

1. [] 基于数据挖掘的服务器日志预警方法研究 王栋;张云龙;唐 琨; -《价值工程》- 2020-01-15 [↑](#endnote-ref-0)
2. [] 大数据环境下机器学习的基本知识点 侯锐; -《区域治理》-

   2019-08-16 [↑](#endnote-ref-1)
3. [] 基于对抗生成网络的不均衡数据分类学习 程企尧 -《大学生论文联合比对库》- 2019-05-20 [↑](#endnote-ref-2)
4. [] 基于深度学习的内部威胁检测方法研究 田原 -《北京工业大学硕士论文》- 2019-05-01 [↑](#endnote-ref-3)