



阜陽師範大學  
FUYANG NORMAL UNIVERSITY

# 本科毕业论文（设计）

题 目： 聚类算法在数据挖掘中的应用

学 生： 杨继业

学 号： 2020120670

学 院： 计算机与信息工程学院

专 业： 计算机科学与技术


入学时间： 2020 年 9 月 1 日

指导教师： 刘淑影 职称： 讲师

完成日期： 2022 年 4 月 29 日

# 诚信承诺

我谨在此承诺：本人所写的毕业论文《聚类算法在数据挖掘中的应用》均系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释，若有不实，后果由本人承担。

承诺人（签名）：

2022 年 4 月 29 日

# 聚类算法在数据挖掘中的应用

**摘要：**随着最近十年来互联网信息的指数型增长，用户查找和利用信息以及内容提供商对文档进行分类和索引变得越来越困难。传统的网络搜索引擎通常会返回数百或数千个搜索结果，用户所花的浏览时间越来越长。在线图书馆、搜索引擎和其他大型文档存储数据库增长如此之快，以至于手动分类每个文档既困难又昂贵。为了解决这些问题，研究人员寻求使用主题文档的自动化方法，以便在最少的人工干预下更轻松地浏览、组织和索引它们。聚类和分类成为机器学习研究的有用且活跃的领域，并且有望克服这个问题。文档聚类被称为无监督和自动将文本文档组织成有意义的集群或组，也就是说，一组文档中的数据共享相同的主题，不同组文档中的数据代表不同的主题。它与分类不同，因为没有使用标记文档的训练阶段。聚类算法因其在数据挖掘、信息获取、查找内容和拓扑分析等领域被广泛研究。本文通过引入一些常见的聚类算法原理和代码逻辑、以及在数学建模中的实际应用，从而论证了聚类算法在数据挖掘中的应用前景以及地位。

**关键词：**划分聚类；基于密度的聚类；层次化聚类；新的聚类方法；数据挖掘

## Application of Clustering Algorithm in Data Mining

**Abstract:** With the exponential growth of Internet information over the last decade, it has become increasingly difficult for users to find and utilize information and content providers to classify and index documents. Traditional web search engines often return hundreds or thousands of search results, and users are taking longer and longer to browse. Online libraries, search engines, and other large document storage databases have grown so rapidly that manually classifying each document is difficult and expensive. To address these issues, researchers seek to use automated methods of subject documents to more easily browse, organize, and index them with minimal human intervention. Clustering and classification have become useful and active areas of machine learning research and are expected to overcome this problem. Document clustering is known as the unsupervised and automatic organization of text documents into meaningful clusters or groups, that is, data in a group of documents share the same topic, and data in different groups of documents represent different topics. It is different from classification because there is no training phase using labeled documents. Clustering algorithms have been widely studied in the fields of data mining, information acquisition, finding content and topology analysis. This paper demonstrates the application prospect and status of clustering algorithm in data mining by introducing some common clustering algorithm principles and code logic, as well as its practical application in mathematical modeling.

**Key words:** K-Means; DBSCAN; Agglomerative; New Algorithm; Data Mining

## 目 录

1 绪论.....	1
1.1 研究背景 .....	1
1.2 研究意义及目的 .....	2
2 主流的聚类算法 .....	2
2.1 划分聚类 (K-Means) .....	2
2.1.1 K-Means 算法的基本原理 .....	2
2.1.2 K-Means 算法的代码实现 .....	2
2.2 基于密度聚类 (DBSCAN) .....	4
2.2.1 DBSCAN 算法的基本原理 .....	4
2.2.2 DBSCAN 算法的代码实现 .....	5
2.3 层次化聚类 (Agglomerative) .....	5
2.3.1 Agglomerative 算法的基本原理 .....	6
2.3.2 Agglomerative 算法的代码实现 .....	6
2.4 新的聚类方法 (IsolationForest) .....	7
3 聚类算法在 2021 年数学建模五一赛 C 题中的应用.....	7
3.1 问题重述.....	7
3.1.1 背景知识 .....	7
3.1.2 具体问题 .....	7
3.2 问题分析.....	8
3.2.1 研究现象综述 .....	8
3.2.2 对问题的具体分析 .....	8
3.3 模型的假设.....	8
3.4 相关定义.....	8
3.4.1 名词解释 .....	8
3.4.2 符号说明 .....	8
3.5 模型的建立与求解.....	9
3.5.1 问题一的分析与求解 .....	9
3.5.2 问题二的分析与求解 .....	13
3.5.3 问题三的分析与求解 .....	16
3.6 模型的改进与推广 .....	17
4 现有聚类算法存在的不足与改进.....	17
4.1 K-Means 算法的缺点 .....	17
4.2 K-Means 算法的改进 .....	18
4.3 DBSCAN 算法的缺点 .....	21
4.4 DBSCAN 算法的改进.....	21
4.5 Agglomerative 算法的缺点 .....	22
4.6 Agglomerative 算法的改进.....	22
5 总结 .....	24
参考文献.....	25
致谢 .....	26

## 1 绪论

数据挖掘（图 1-1 所示）的标准流程一般从定义问题到建立数据库，再到运用工具分析数据，再到数据准备，最后拿着前面已经处理好的数据进行建立模型，模型一旦建立好，我们就可以进行评估与部署了。数据挖掘一直是研究信息和商业分析的活跃领域（如图 1-2）。现有的数据挖掘技术包括特征、分聚类、关联等。而聚类算法在数据挖掘中最重要的一个应用就是对用户进行聚类——找到信息较为相识的用户群体，例如访问相似页面的用户。通过分析集群的特征，程序员们从而可以更好地了解用户，通过大数据技术，可以为用户提供更合适的个人定制、消息推送服务。本文主要介绍一些当下主流的聚类算法工作原理与代码实现，以及他们在数据挖掘中实际应用。

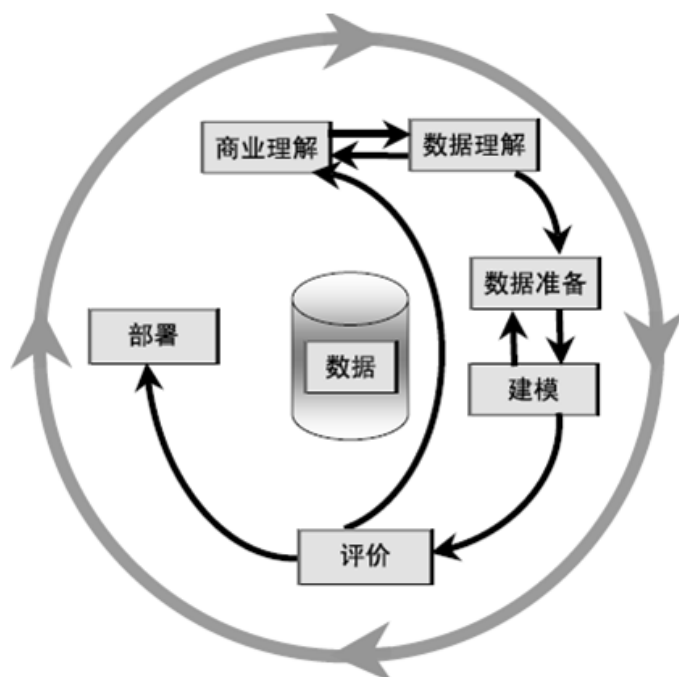


图 1-1 数据挖掘的标准流程：CRISP-DM 模型

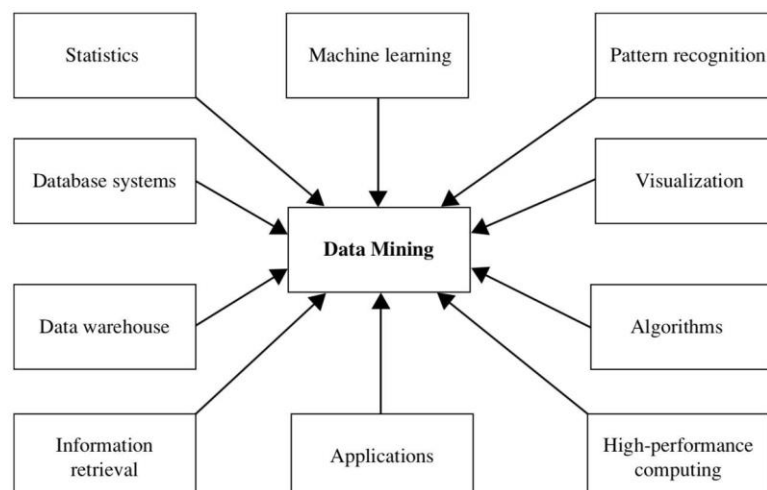


图 1-2 数据挖掘的活跃领域

## 1.1 研究背景

随着 Computer Science、Network Changes、Internet 的快速发展，互联网中大量的信息的堆积，导致个人、政府、商业群体在这些巨大的信息中，找到有用的、有价值的信息变得异常困难。使用编程、数据分析工具、智能化机器等，迅速地找到有价值的信息迫在眉睫。此外，用于开展数据挖掘所使用的数据源必须真实且庞大。被发现的信息和知识对用户来说应该很有趣并且有用，且具有一定的价值。通常来说，数据挖掘的结果不需要完全正确的知识，而是用它所得到的结论去预测一种趋势。传统数据库（Mysql、SQL Sever 等）诞生以来，人们就期望着把数据挖掘与数据存储技术结合到一起，以便用于存储、分析网络中的这些非结构化的数据。只是当前网络中存在着大量的数据，而网络中的文本数据存在着高维、空间分布稀疏的特点，就使得目前的数据挖掘算法在挖掘网络文本海量数据过程中普遍存在挖掘精度低、耗时长等问题。

## 1.2 研究意义及目的

今天，互联网技术、云计算技术、信息技术等各种数据和信息技术飞速发展。随着各种产业的快速发展，信息化成为社会和经济发展的主要发展趋势和经济增长的重要内容。在这个时代，计算机科学家对这些问题做了很多研究。人们提出了各种大量的数据挖掘算法，其中聚类算法既可以对数据进行有意义的组合，也可以进行划分，将来自类似对象的数据拆分成类，不同对象的数据可以分割成不同类别。我们把这些数据构建起它们的多层数据间与数据间的回归向量空间上的模型，从而找到它们的特征从而完成数据的聚类，也就利用数据聚类结果完成数据挖掘。正是由于这种可靠的算法的出现，既解决了大量的不同数据归类问题，又节约了时间成本，有效的提升了数据挖掘的效率，并提升了可靠性，对于未来数据挖掘中归类问题发挥着重要的作用，使得其广泛的应用于数学建模与数据挖掘中。

## 2 主流聚类算法

### 2.1 划分聚类（K-Means）

**K-Means** 通常指的是无监督的聚类算法，什么是“无监督”呢？，“无监督”说的是用来训练样本数据的标记信息开始的时候是不知道的，通过让机器用代码去学习一开始没有标记的训练样本，从而得到数据规律与属性，最后为我们想用聚类解决某个问题提供有力的的数据分析基础。因此 **K-Means** 实施起来很简单，聚合效果好，被广泛使用。

### 2.1.1 K-Means 算法的基本原理

**K-Means** 算法的工作原理其实非常简单：一开始我们要确定好常数  $k$ ， $k$  代表要分成几类。然后随机选取某个点作为初始的点，而这个点正是质心，计算数据中的每一个点与刚才选取的质心相似程度，获得我们的相似程度数据，那么相似度是什么呢？其实就是数学上的大名鼎鼎的欧氏距离，有了这些用作参考的数据，然后再将获得的样本点分类为最相似的类别，然后通过计算确定了类的中心，一直重复下去。直到跟踪条目保持不变。最后，找到它们所在的类别（每个采样点）和每个类别的中心。

### 2.1.2 K-Means 算法的代码实现

我们使用 **python** 实现算法：①选择  $k$  个簇的数目；②根据新划分的群集更新“群集中心”；③根据新划分的集群更新“集群中心”。④可视化展示，利用 **python** 第三方库中的 **matplotlib.pyplot** 对聚类后的元素显示（散点图），方便查看结果。代码请见附件 **DataKMeans.py**，由聚类算法聚类结果把数据划分成如下图 2-1：

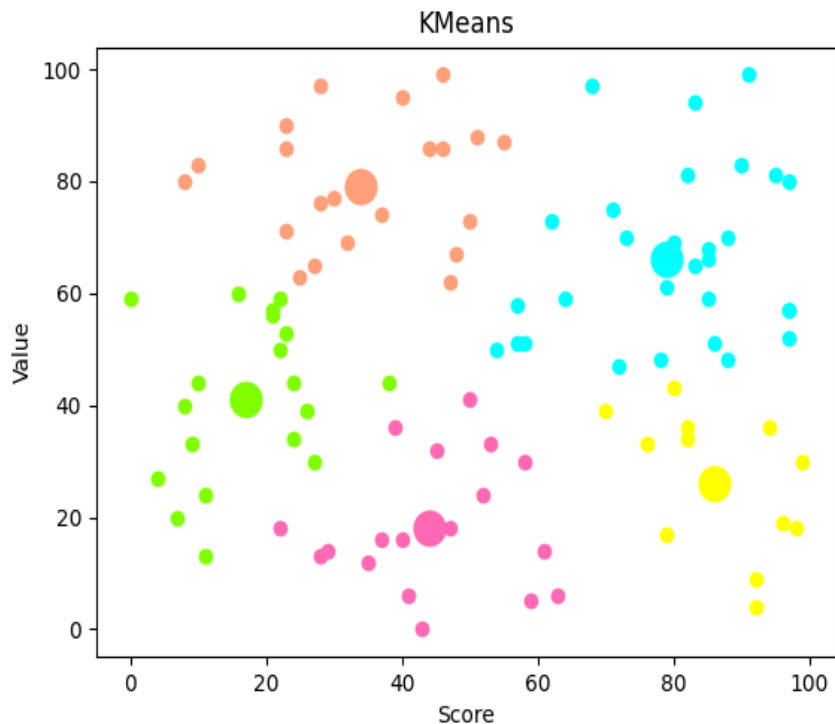


图 2-1 随机生成的散点数据被聚类算法成功划分成若干组

## 2.2 基于密度聚类 (DBSCAN)

DBSCAN 不同于其他聚类方法，它是一种基于密度的聚类方法。在处理不同密度、大小和形状的空间集群时，检测点集群可能具有挑战性。如果数据包含噪声和异常值，任务可能会更加复杂，这与分区和层次聚类算法不同。为了处理大型空间数据库，科学家们发明了 DBSCAN。

### 2.2.1 DBSCAN 算法的基本原理

DBSCAN 算法使用非常简单的群集概念。最终集群的一个集群由一组可以从密度到关系的最大密度链样本组成。一个或多个关键点可能存在于 DBSCAN 算法群集中。如果有中心点，则群集中其他非核心点的示例位于核心点的 EPS（扫描半径）内。如果有多个关键点，则群集中每个关键点的 EPS（扫描半径）附近必须有其他关键点。一个 DBSCAN 集群是由这些核心位置的 EPS（扫描半径）中存在的所有数据的集合组成的。因此 DBSCAN 能够计算随机事件发生的概率。DBSCAN 算法的描述如：①输入：问题中的数据源的数据对象数、EPS 和 Minpts。②输出：密度高的集群用输出作为连接，处理过程如下：（1）任意点 P；如果 P 是 EPS 和 Minpts 参数的中心点，就能找到 P 的数据对象的密度访问点，从而构建集群。如果指定数据对象的 P 点是边界点，则选择其他数据。重复步骤 2 和步骤 3，一直到解决所有项目。

这里值得注意的是：①Epsilon ( $\epsilon$ )代表了社区的最大半径。指定了数据点之间的距离  $\epsilon$  以下的情况下，它们属于同一类。也就是说，该算法具有两个相似点，可以拿来衡量同一类别的距离。大的社区半径生成大的集群，小的社区半径生成小的集群。通常，我们喜欢小的值，因为我们只需要在它们之间的距离内的少量数据点。然而，如果它太小，则集群被划分成更小的集群。②Minpts：核心点的最小邻域。较低的 MINPT 对于算法构建具有更多噪声或异常值的集群是非常有用的。最小值越高，集群就越结实，同事集群也不能太大，否则一旦集群超过预估范围，小的集群将被吞并到大的集群中去。

### 2.2.2 DBSCAN 算法的代码实现

我们使用 python 实现该算法：①随机选取点。就是那条距离能够找到所有的点。如果起始 EPS 中的数据点数小于最小示例数，则该点将显示为噪声。如果 EPS 中的数据点数超过最小示例数，则会将该点指定为核心示例，并指定新的群集标签。②把点移动到邻居上（距离的 eps 以内）。如果未分配组，请转发新创建的组。如果他们是核心样本，请逐个拜访他们的邻居。由于集群的稳定发展，核心样本不存在于集群的 eps 距离中。③选择未访问的其他点，并重复步骤。④EPS 设置较低时，点中的任何一个都不会被视为核心样本。而且这也可以把所有的点都标记为噪声点。⑤EPS 设置为非常大的值。虽然不需要指定捆扎的数量，但是设置 eps 的话，eps 的数量可以暗中调节。⑥使用 StandardScaler 或 MinMaxScaler 缩放数据；详情请见材料中的 DBSCAN-Data.py，经过 DBSCAN 算法，在不同参数下聚类，结果如图 2-2：



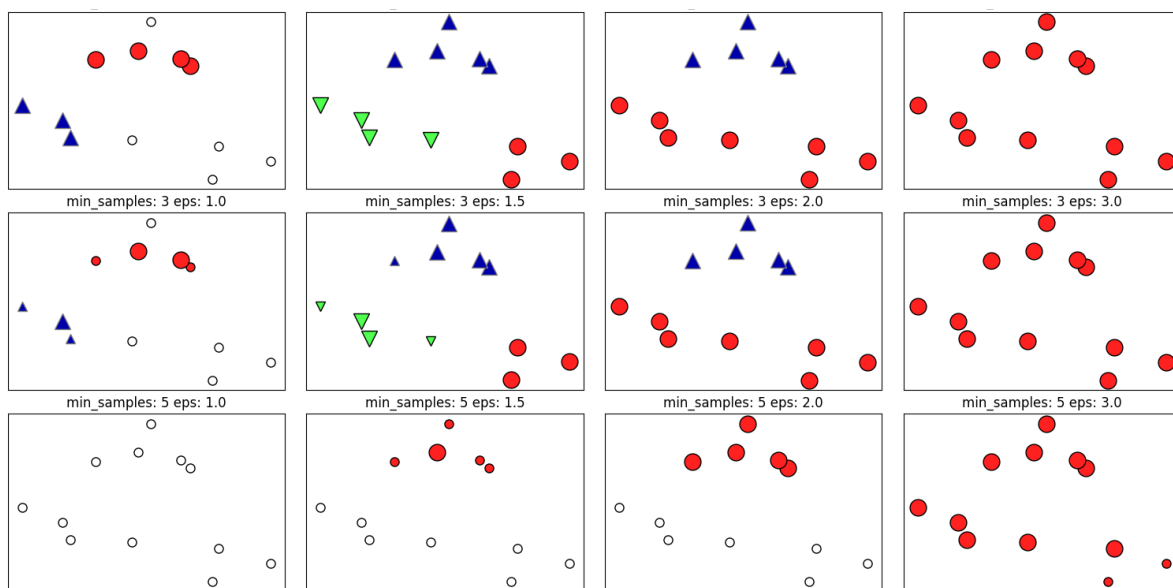


图 2-2 密度聚类算法在不同参数下对数据聚类成若干组的影响

### 2.3 层次化聚类 (Agglomerative)

分层分组是非常直观的算法。正如名称所示，小集群需要从下到上分组，并且大集群需要从上到下分割。一般用于从下到上的汇总。从底部到上方的聚类融合是指一次发现两个短的聚类，将它们结合成大的聚类，一直到所有的类都被我们聚成一个大的类。整个过程构建树结构。如图 2-3:

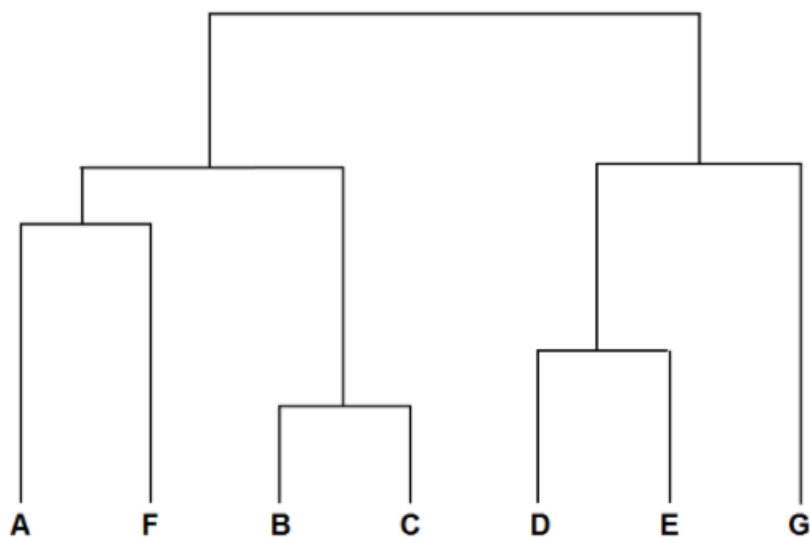


图 2-3 层次聚类算法树型结构图

#### 2.3.1 Agglomerative 算法的基本原理

这就是分层类合并算法所使用的方法就是通过计算两种类型数据与数据之间的相似程度，将所有数据点中相似的两个数据点相结合，频繁地重复此处理。然后分层合并

处理计算了每种类型的数据点和所有数据点之间的距离，并识别出它们的相似性。相似程度越大，距离越短。所以说要创建聚类树，首先要做的是连接两个最近的数据点或类别。

一开始层次聚类要把数据中的每一个对象视为一个簇，然后再通他们之间的距离，来合并最小距离的两个簇，也就是说每次结果的总簇数将减一，直到只有一个簇为止。

### 2.3.2 Agglomerative 算法的代码实现

利用 python 我们能够很好的实现 Agglomerative 算法：①考虑到每一个簇对象，算出簇与簇的距离。②现存两个距离值最小的簇重新组合成新的簇。③再次计算新的与旧的距离。④重复这个过程，最后只剩一个大簇。代码请见附件 AgglomerativeData.py，聚类结果如图 2-4：

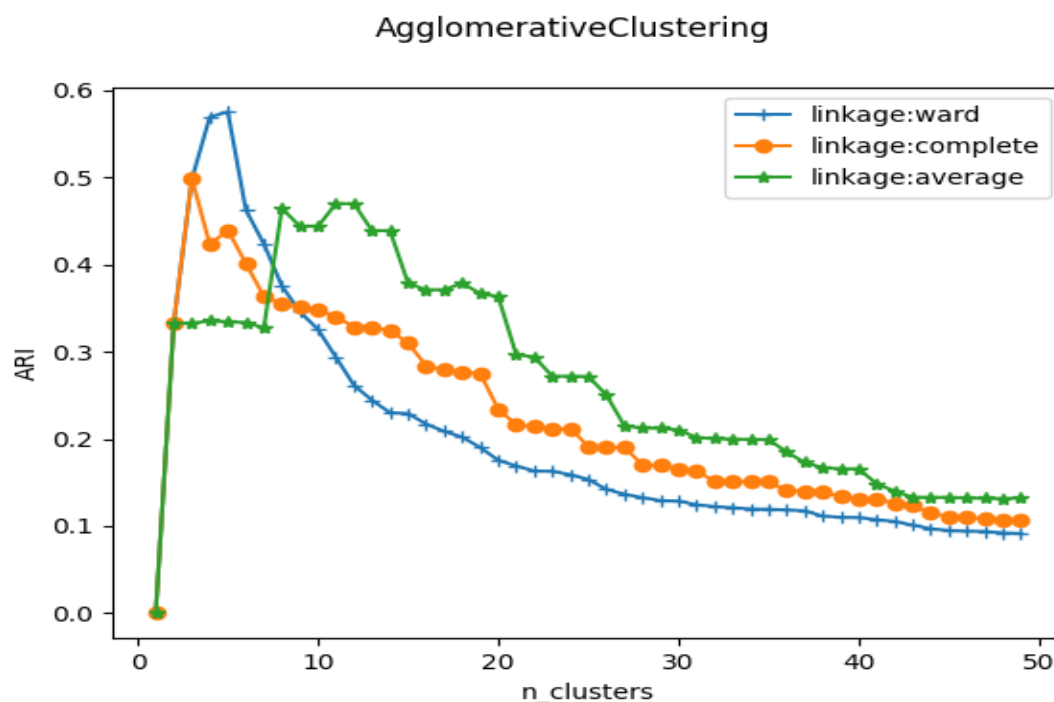


图 2-4 层次聚类把数据聚类成 3 组

### 2.4 新的聚类方法 (IsolationForest)

异常检测是发现偏离规范的数据点。换句话说，这些点不符合预期模式。异常值和异常是用于描述异常数据的术语。异常检测在各个领域都很重要，因为它提供了有价值且可操作的见解。例如，核磁共振成像扫描中的异常可能表明大脑中存在肿瘤区域，而制造厂传感器的异常读数可能表明组件损坏。异常值只是一个数据点，它与特定数据集中的其余数据点有很大的偏差。同样，异常检测是帮助我们识别数据异常值或与大量其他数据点有很大偏差的点的过程。当涉及到大型数据集时，可能包含非常复杂的模式，这些模式无法通过简单地查看数据来检测。因此，为了实现关键的机器学习应用，异常检测的研究具有重要意义。

就像随机森林一样，隔离森林也是使用决策树构建的。它们以无监督的方式实现，因为没有预定义的标签。隔离森林的设计理念是异常是数据集中“少数且不同”的数据

点。决策树是使用诸如基尼指数或熵之类的信息标准构建的。明显不同的组在树的根部被分开，在树枝的更深处，更细微的区别被识别出来。基于随机挑选的特征，隔离森林以树状结构处理随机二次采样的数据。深入到树中并需要更多切割来分离它们的样本很少有可能是异常的。同样，在树的较短分支上发现的样本更有可能是异常，因为树发现更容易将它们与其他数据区分开来。本文将会在第三章中具体应用这一聚类算法完成数据分析。

### 3 聚类算法在 2021 年数学建模五一赛 C 题中的应用

#### 3.1 问题重述

##### 3.1.1 背景知识

##### 1. 背景介绍

新时代背景下，国内安全事故整体下降，许多企业稳定生产，有好转的趋势，只是现存的形式不是那么的乐观。事故总数依旧很高。一些企业为了利益，违法生产严重，严重事故频发。为了国家的繁荣稳定也为了进一步加强安全生产工作，全面提高企业安全生产水平刻不容缓，运用数学知识来根据生产过程中产生的数据帮助企业用以建立模型最大化，从而减少风险事故发生情况。

##### 2. 问题的产生

在企业生产过程中，可能有某些潜在的风险无法被轻易发现，但是制造过程中经常出现的数据可以实时反映这些潜在的风险，收集到的生产数据可能会或多或少地发生变化。这些数据波动的一部分是由于外部温度和功率的变化而引起的正常波动，不发生安全隐患，不需要人类干预的异常波动存在潜在的安全隐患，表明生产过程是由不稳定因素引起的，被视为危险异常，因此生产安全的问题也就由此产生。

##### 3.1.2 具体问题

##### 1. 问题一

根据附件中经过脱敏处理后的数据，对异常数据进行过滤，建立异常风险数据和异常风险数据编号的识别模型。

##### 2. 问题二

结合问题一的处理结果，按百分制进行异常程度评价，找到异常数据填入表中。

##### 3. 问题三

结合问题 2 提出的风险异常度定量评价方法，建立了风险预警模型，预测了 23:00 至 23:59 可能存在的风险性异常，填入数据。

#### 3.2 问题分析

为了更好的进行数据挖掘，我们进一步分析问题。

##### 3.2.1 对问题的具体分析

##### 1. 对问题一的分析

由于附件 1 所给出的已经进行数据脱敏的时间序列数据，因此我们选择利用箱型图

对附件 1 中的数据进行异常值筛选，导出非风险性异常数据和风险性异常数据，并且利用 MDS 多维标度法进行数据处理，使用 python 进一步处理，转化为二维数据并绘制成图，最后通过异常点检测算法 Isolation Forest 来判定非风险性异常数据和风险性异常数据。

### 2. 对问题二的分析

结合问题一的结果，用我们已经建立好的数学模型，将数据归一化，得到的结果为 [0,1]，分析处理后越接近 1 表示异常程度越高，越接近 0 则表示异常程度越低，再将其最佳转化百分制表示，筛选出得分最高的 5 个，对比后找到最高的 5 个时刻及这 5 个时刻对应的异常传感器编号填入表中。

### 3. 对问题三的分析

SARIMAX 模型适用于本题，再结合问题二的处理结果就能建立风险性异常预警模型，预测出 23:00:00-23:59:59 中四个时间段的最高异常分值及对应的异常传感器编号，填入表中。

#### 3.2 模型的假设

1. 假设在生产过程中设备不会出现故障。
2. 假设设备都处于正常寿命使用期。
3. 假设设备工作的地理位置不包括中国东三省（零下几十度）。

#### 3.3 相关定义

##### 3.3.1 名词解释

##### 1. 箱型图

箱型图，数学问题中常见的图像，专门用来查看数据的分布特征。

##### 2. 多维标度法（MDS）

MDS 是一种多元数据分析技术，就是降高维数据转化成地维，同时还能够保持数据相识度。

##### 3. 孤立森林算法

孤立森林算法对于检测连续数据中的异常很有用，它将异常定义为“容易隔离的异常点”，也可以理解为从高密度组中分离出来的具有稀疏分布的点，可以区分异常点。

##### 3.4.2 符号说明

我们给出如下表的符号说明：

表 3.4.2 符号说明

符号	说明
$Q_3$	箱型图中的上四分位数
$X_m$	箱型图中的数据均值
$Q_1$	箱型图中的下四分位数
$D_{ij}$	原始空间下的距离阵

$X_i$	空间中第 <i>i</i> 个点
$H(x)$	叶子节点到根节点的路径长度
$S_{(x,n)}$	记录 <i>x</i> 在 <i>n</i> 个样本中的训练数据构成的孤立树的异常指数
$Y_{SC}$	原始数据中的 Score

### 3.4 模型的建立与求解

#### 3.4.1 问题一的分析与求解

##### 1. 对问题一的分析

这个问题需要我们对附件 1 中的数据进行异常值筛选，区分非风险异常数据和风险异常数据。我们从原始数据中过滤掉所有异常数据，然后再对其进行处理以将其转换为二维数据并进行绘图。最后，利用异常点检测方法来区分非风险数据和风险异常数据。

##### 2. 对问题一的求解

我们把数据可视化成箱型图，通过观察箱型图找到异常。箱线图，通常称为箱线图，箱型图由六个部分组成，专门用来可视化数据与数据间的离散程度。对于数据，确定他的上边缘、上四分位数  $Q_3$ 、中位数  $X_m$ 、下四分位数  $Q_1$ 、下边缘和异常值。我们根据这个定义生成箱线图并检测异常数据。异常值通常被定义成为小于  $Q_1 - 1.5IQR$  或大于  $Q_3 + 1.5IQR$  的值，因此我们绘制该图形来确认异常数据，如图 3-1 所示：

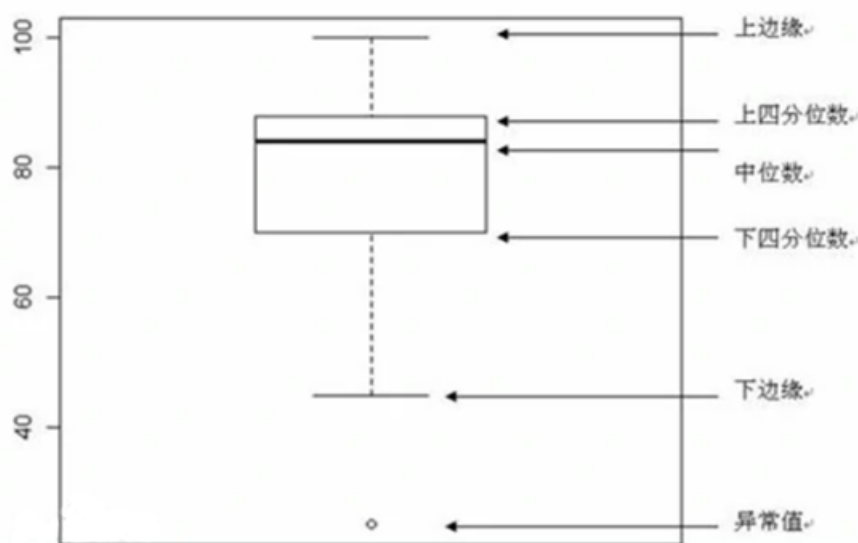
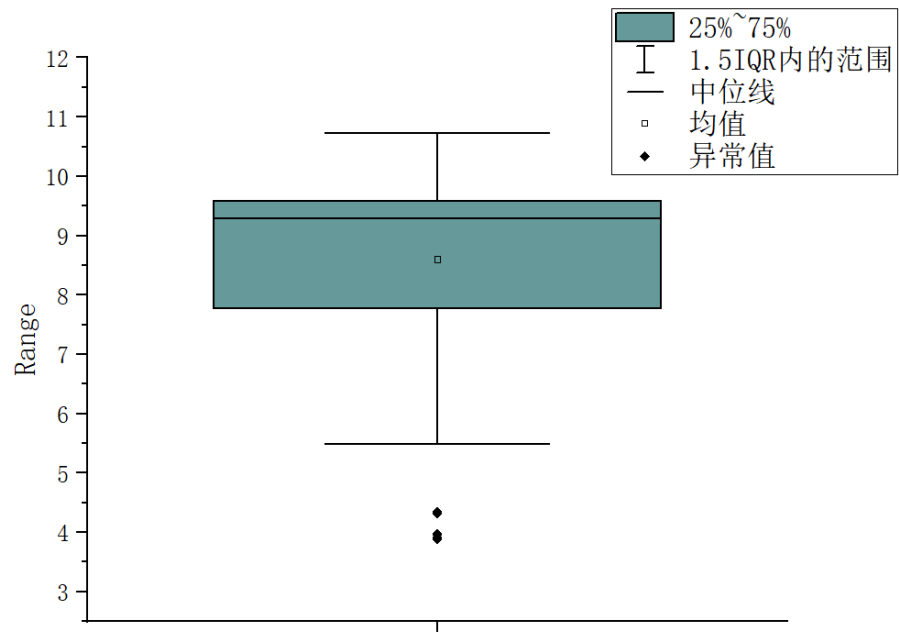


图 3-1 箱型图示意图

由此，我们先对题设所给数据进行异常值筛选，由于数据项编号共有 100 项，列举全部箱型图过于繁琐，我们只展示出编号为 1 的数据的箱型图，后续余项箱型图不再一一列举。以编号 1 数据为例，绘制出编号一的箱型图如下图 3-2 所示：



传感器编号为1的箱线图

图 3-2 编号 1 数据的箱型图

在图 6 中我们可以看出编号为 1 数据的上四分位数 $Q_3$ 、中位数 $X_m$ 、下四分位数 $Q_1$ 、下边缘、和该数据的异常值，我们记录该数据所有的异常值。以此类推，画出编号为 1-100 的所有数据的箱型图，记录所有的异常数据。

在收集所有异常数据后，我们在第二阶段使用 MDS 多维缩放方法将所有异常数据转换为二维数据。这是因为太多的维度使得我们不能很好的找到特征数据，数据与数据间的属性、联系不是那么的好，所以我们要把高维降成低维，便于我们找到特征以及处理数据。

由于原空间中的距离矩阵和低维空间中的距离矩阵假定为欧氏距离矩阵，因此距离矩阵  $D$  为欧氏距离矩阵，即存在一个正整数 $R_p$ 和 $n$ 点 $x_1, x_2, \dots, x_{100}$ ，使得有：

$$d_{ij}^2 = \|x_i - x_j\|^2, i, j = 1, 2, 3 \dots 100$$

该公式称为 Classical MDS 算法公式，作用是用来寻找  $D$  的（拟合）构图。经过 MDS 多维标度法处理后的编号 1-100 的数据如下图所示：

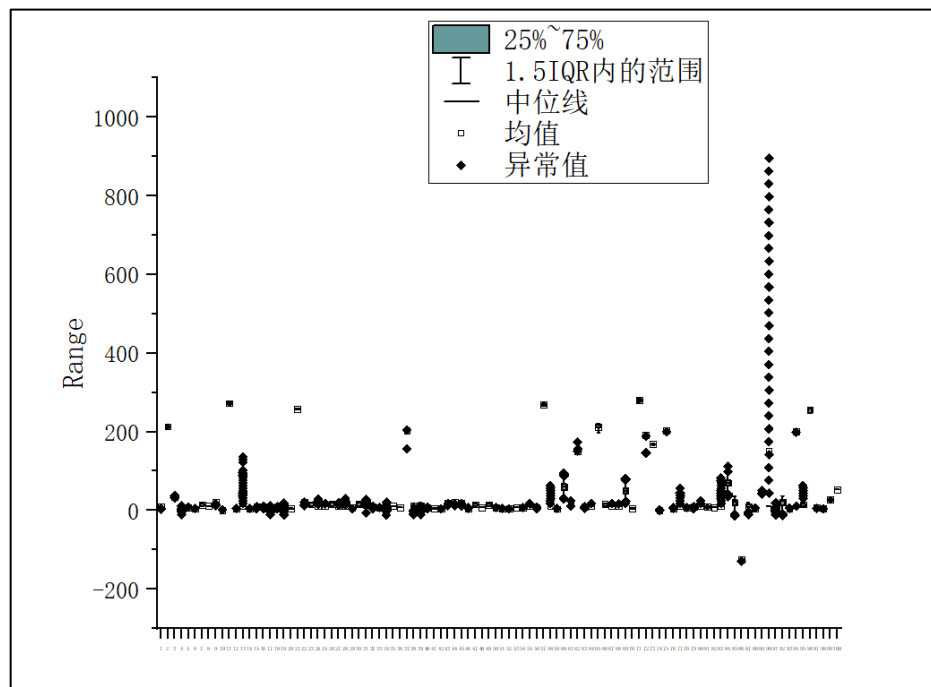


图 3-3 编号 1-100 数据的异常数据

由此我们得到所有的异常数据，在最后一步，我们选择异常检测算法，即孤立林算法来区分所有异常数据中的风险异常。孤立森林像差检测算法的目标是寻找与大多数数据集不同的数据。一般情况下，在数据预处理过程中，可以对异常数据进行过滤，避免归一化等处理结果。它还可用于过滤特征数据，而无需标记输出来检测异常数据。

随机森林算法可用于使用数据检测连续数值数据的异常。以上描述为“分离可能性很高”。而且这可能意味着“稀疏的分散”和“远离密集的团队”。为了使用统计说明，数据空间中的稀疏分散区域指示出现在该位置的数据的概率很低，指示数据进入该位置是很奇怪的。

使用 IsolationForest 算法返回每个样本的异常分数，IsolationForest 通过随机选择一个特征，然后在所选特征的最大值和最小值之间随机选择一个分割值来“隔离”观察结果。由于递归划分可以用树结构表示，因此隔离样本所需的分裂次数等于从根节点到终止节点的路径长度。这个路径长度，在这些随机树的森林上平均，是衡量正常性和我们的决策函数的指标。随机分区为异常产生明显更短的路径。因此，当随机树的森林共同为特定样本产生较短的路径长度时，它们很可能是异常的。

在此之后，我们利用 python 对数据进行编写处理，数据处理代码（部分）如下图 3-4 所示：

```

X_train = a
outliers_fraction = 0.05
n_samples = len(a)
# 构造模型并拟合
clf = IsolationForest(max_samples=n_samples,
                      random_state=rng,
                      contamination=outliers_fraction)

clf.fit(X_train)
# 计算得分并设置阈值
scores_pred = clf.decision_function(X_train)
f = open('Score.txt', 'w')

```

图 3-4 python 处理数据代码

最终所得 Isolation Forest 示意图如下图 3-5 所示：

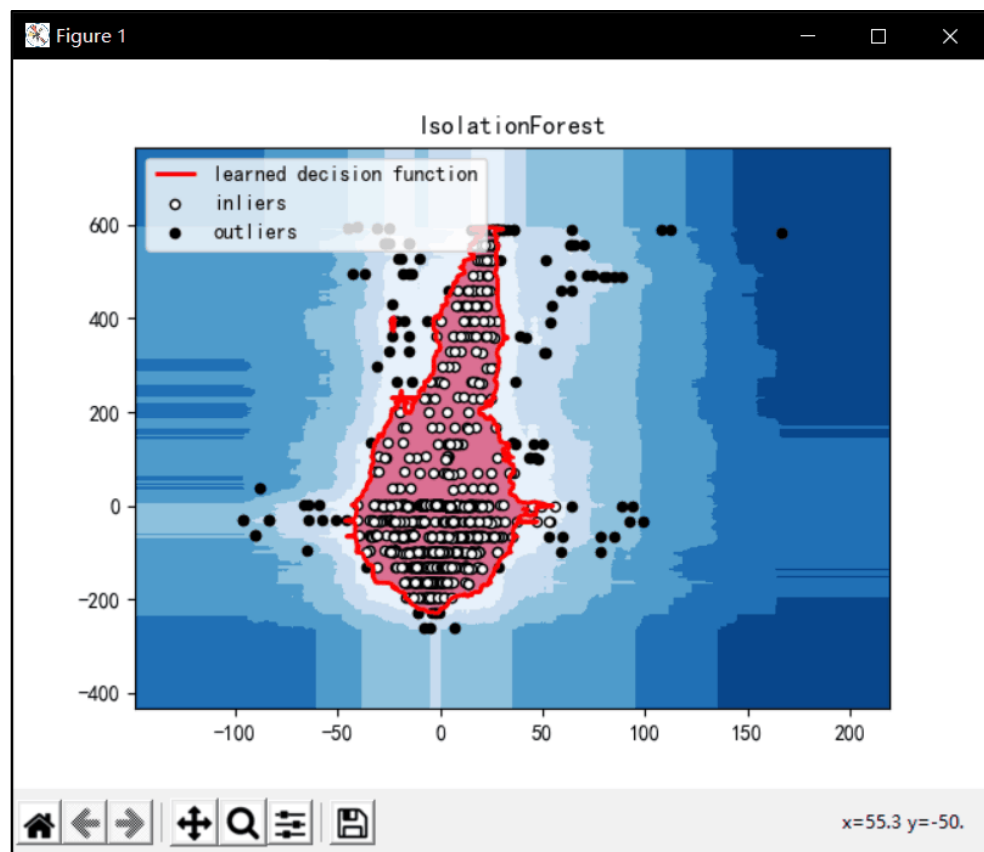


图 3-5 Isolation Forest 示意图

根据图 9 所示，红色区域为孤立森林决策树的学习决策函数边界，可以用来区分风险性异常数据和非风险型异常数据，白色的点表示非风险性异常数据，黑色的点表示离群点，也即风险性异常数据。综上，通过模型的建立我们给出了判断非风险性异常数据和风险性异常数据的方法。接着我们验证该模型的准确性，首先我们测试模型的泛化能力，



我们对模型已经输出的数据可视化，如图 3-5-1：

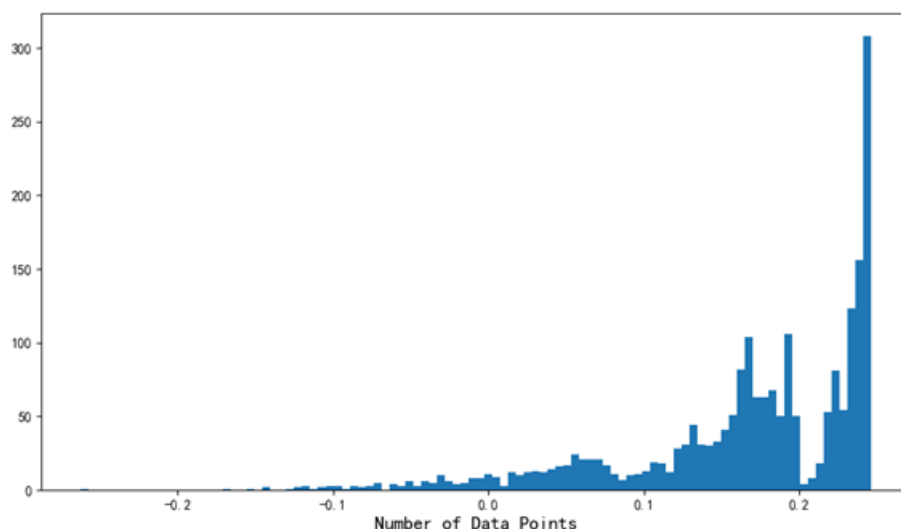


图 3-5-1 模型输出的数据示意图

我们可以看到大部分数据的值都是较高的，而异常值恰恰就是图中值较小的，假设我们认定小于-0.13 的是异常值，那么我们对模型已经输出的数据在进行 AUC 评价，如图 3-5-2，最后得到如图 3-5-3 结果，可见我们的模型效果以及泛化能力还是不错的。

```
anomalies = anomaly_scores > -0.13

# print('the shape of anomalies and matches')
auc = roc_auc_score(anomalies, matches)

print("AUC:{:.2%}".format(auc))
```

图 3-5-2 模型 AUC 评价代码图

```
...
[ 4.79679463e+00 -3.55258794e-
[ 4.79679463e+00 -3.55258794e-
[ 4.79679463e+00 -3.55258794e-
AUC:92.00%
```

图 3-5-3 模型 AUC 评价结果图

### 3.4.2 问题二的分析与求解

#### 1.对问题二的分析

题设需要我们以百分制构建出合适的数学模型，并且给出衡量异常程度的评价方法，我们已经构建好孤立森林模型，只需要把模型得出的数据进一步对数据进行归一化处理，最后得到的结果为[0,1]，越接近 1 表示这个数据异常度可能高，越接近 0 代表该数据异常度可能低，再利用代码求出最佳转化百分制的函数，筛选出得分最高的 5 个，

对比之后我们找到那五个异常程度最高的填入表中。

## 2.对问题二的求解

为了更好地定量评价异常风险数据的异常程度，我们对孤立林中的孤立树进行了分析和处理，孤立树是一个随机二叉树，每个节点有两个子节点或叶子。

构建孤立森林后，可以导入数据，即在隔离树上记录数据，并查看数据记录在哪个叶节点上。从一棵孤立的树上实际上可以检测到异常值的原因是，异常值通常很少，并且可以归因于一个划分较小的单独区域，即异常值将在一棵孤立的树上快速划分为叶节点。

所以，我们求出从叶节点离根节点的路径距离长度 $H(x)$ ，以评估数据集  $x$  是否是离群值（即，根据 $H(x)$ 的值来判断 $x$ 是否是异常点）。例如 100 个数据集集中的数据，需要构建树的最小高度为 $\log(100)$ ，最大高度为 99。所需标准化公式如下：

$$S(x, n) = 2^{-\frac{H(x)}{C(n)}}$$

$$C(n) = 2H(n - 1) - (2(n - 1)/n)$$

$$H(k) = \ln(k) + \tau, \quad \tau = 0.5772156649$$

$S(x, n)$  代表 $x$ 在 $n$ 个样本的测试中的数据组成的孤立树的异常指数， $S(x, n)$ 有一定的取值范围，取值的范围为 $[0,1]$ ，如何判断异常如下表 5.1 所示：

**表 5.1 孤立树异常程度判断**

意义	
$H(x) \rightarrow 1$	$H(x)$ 越接近 1 表示异常程度越高
$H(x) \rightarrow 0$	$H(x)$ 越接近 0 表示异常程度越低
$H(x) \rightarrow 0.5$	$H(x)$ 约等于 0.5 则说明数据集基本上没有的异常值

对数据进行归一化处理后的结果（部分）如下图 3-6 所示：

BU	BV	BW	BX
70	71	Type	Score
4.062556	3.840573	-1	-0.26226
4.076404	3.840573	-1	-0.15244
4.127178	3.840573	-1	-0.14207
4.127178	3.840573	-1	-0.08825
4.127178	3.840573	-1	-0.12513
4.436441	3.840573	-1	-0.11719
4.436441	3.840573	-1	-0.07164
4.159489	3.840573	-1	-0.07686
4.164105	3.840573	-1	-0.03741
4.13641	3.840573	-1	-0.03023
4.13641	3.840573	-1	-0.0305
4.131794	3.840573	-1	-0.04661
4.145642	3.840573	-1	-0.0774

图 3-6 对数据进行归一化处理结果（部分）

在得到数据归一化处理结果后，经过代码处理将其转化为百分制（分值越高表示异常程度越高），代码如下图 3-7 所示：

```
# print(data['Score'].sort_values())
# data['Score']=(data['Score']-0.73)*(-100)
# data.to_csv('2—完整.csv',encoding='gbk')
# (data.sort_values('Score').iloc[-5:]).to_csv('5条.csv',encoding='gbk')
```

图 3-7 将归一化结果转化为百分制

由图 5.7 可知，Score 中数据转化为百分制（分值越高表示异常程度越高）的最佳公式为：

$$Y = Y_{SC} * (-0.73) * 100$$

转化过的结果如下图 3-8 所示：

	BU	BV	BW
0	71	Type	Score
6	3.840573	-1	99.22601
4	3.840573	-1	88.24358
8	3.840573	-1	87.20745
8	3.840573	-1	81.82526
8	3.840573	-1	85.51333
L	3.840573	-1	84.71936
L	3.840573	-1	80.16352
9	3.840573	-1	80.68571
5	3.840573	-1	76.74122

图 3-8 转化百分制

我们筛选出得分最高的 5 组数据，也即数据中异常分值最高的 5 个，与原数据进行对比，找到这五组数据所对应 5 个时刻及这 5 个时刻对应的异常传感器编号，对比代码（部分）如下图 3-9 所示：

```

ind = []
Name = []
for i in data.values:
    for j in i:
        j = j.round(3)

        boolNum = (j==originData.values)
        for x in range(len(boolNum)):
            for k in range(len(boolNum[x])):
                if True == boolNum[x][k]:
                    Name.append(k)
                    ind.append(x)

```

图 3-9 转化后数据与原数据对比代码（部分）

将最终结果填入表 1，如下表 1 所示：

表 1 问题 2 的结果

	第一高分	第二高分	第三高分	第四高分	第五高分
异常程度得分	99.22601	89.6219	88.24358	87.24175	87.24175
异常时刻编号	12:38:00	12:38:00	22:32:30	22:32:30	6:13:45
异常传感器编号	63	72	90	53	31
异常传感器编号	54	68	61	51	52
异常传感器编号	44	43	14	48	38
异常传感器编号	100	8	95	97	55
异常传感器编号	22	49	16	42	35

### 3.4.3 问题三的分析与求解

#### 1. 对问题三的分析

这个问题需要建立一个模型来早期发现未来生产过程中的潜在风险。我们选择了合适的 SARIMA 模型，并结合问题 2 处理的结果构建了风险异常警报模型。根据模型处理结果，得到了与 23:00-23:59 4 期间的最高异常点数对应的异常传感器数，并记入表中。

#### 2. 对问题三的求解

问题三需要建立风险性预警模型，我们进行模型对比，发现 SARIMA 较为适合进行模型建立。

我们利用代码对数据进行建模，对模型的建立代码（部分）如图 5.10 所示：

```

#         results = mod.fit()
#         print('x{}12 - AIC:{}'.format(param_seasonal, results.aic))
#         if results.aic < score_aic:
#             score_aic = results.aic
#             params = param_seasonal, results.aic
#         param_seasonal, results.aic = params
#         print('x{}12 - AIC:{}'.format(param_seasonal, results.aic))
#     pdq = [0, 1, 1]
#     get_ARIMA_params(train, pdq, m=10)

|
y_hat_avg = test.copy()
fit1 = sm.tsa.statespace.SARIMAX(train[1],
                                order=(0, 1, 4),
                                seasonal_order=(3, 2, 4, 10),
                                enforce_stationarity=False,
                                enforce_invertibility=False).fit()
y_hat_avg['SARIMA'] = fit1.predict(5279, 5520)
plt.figure(figsize=(16, 8))
plt.plot(train[1], label='Train')
plt.plot(test[1], label='Test')
plt.plot(y_hat_avg['SARIMA'], label='SARIMA')
plt.legend(loc='best')

```

图 3.10 对数据进行分析处理（部分）

输出结果如图 3-11 所示：

```

当前时刻为： 23:15
异常的感应器有：  ['感应器6', '感应器12', '感应器14', '感应器20', '感应器46']
异常得分为：  140.87487118928732

当前时刻为： 23:30
异常的感应器有：  ['感应器6', '感应器12', '感应器14', '感应器20', '感应器46']
异常得分为：  140.51674777337848

当前时刻为： 23:45
异常的感应器有：  ['感应器6', '感应器12', '感应器14', '感应器20', '感应器46']
异常得分为：  140.48243399368258

当前时刻为： 24:00
异常的感应器有：  ['感应器6', '感应器12', '感应器14', '感应器20', '感应器46']
异常得分为：  140.52817104974736

```

图 3-11 输出结果

表 2 问题三结果

时间	23:00:00-23:14:59	23:15:00-23:29:59	23:30:00-23:44:59	23:45:00-23:59:59
异常程度得分	140.8	140.5	140.4	140.5
异常传感器编号	6	6	6	6
异常传感器编号	12	12	12	12
异常传感器编号	14	14	14	14
异常传感器编号	20	20	20	20
异常传感器编号	46	46	46	46

### 3.5 模型的改进与推广

本文在数据处理的过程中，由于时间及能力有限，在某些环节会有数据差异或者误差等等，可能会造成一定程度的结果失真，但从实际结果来看，也可较为完美的反映所

得到的一般规律，可侧面反映出模型的实用价值。

## 4 现有聚类算法存在的不足与改进

### 4.1 K-Means 算法的缺点

(1) K-Means 先确立好聚类成几簇对最后的结果有着重要的影响。在聚类之前，我们需要预先确定 K 的大小，但是很难确定哪个类别是最好的。例如，如果数据集能够被清楚地分为两类，即 K 为 2 是最好的，如果我们聚类的数据非常大，那么我们就很难一开始就确定好 K 的值。

(2) K-Means 聚类效果取决于一个数据对象的噪声与异常。当一个噪声点被添加到数据集中时，它被分成一个类。当然，如果  $K=2$ ，其他点属于同一类，噪声点属于它们自己的类。可以区分的点受噪声影响，成为一个类。对于  $K=3$ ，噪声也属于同一类别，其他数据分为两类。这表明噪声对其他点的分类有很大的影响。

下面我们给出改进思路以及代码实现。

### 4.2 K-Means 算法的改进

**Bi-Kmeans 主要思想：**(1)现有的母簇都有母子分支，从候选集（本簇也是母簇）中选拔误差平方和降低量最大的母簇做分裂，聚类个数增加 1。(2)母簇分裂得到两子簇，取代原来的母簇，正式成为新簇。(3)聚类个数达到要求，则可以提交分裂结果。新簇内部还没有二分聚类，如果聚类个数不够，新簇如循环初始一样，继续二分聚类，成为新生的母簇。(4)每次分裂的两个新簇各自二分聚类，生成两簇母子分支，新生的母簇加入到候选待分裂的母簇集中，继续开头所描述的过程。

循环过程切入初始条件：①初始：单一簇整个为新簇。②新生成的簇切入初始条件，内部二分割聚类，变成新生的母簇。初始的母簇是单一的。③母簇概念的明确：具备母子分支，又不分裂，才能称之为母簇。获取各母簇的母子分支的误差平方和降低量（在前一步新簇变为母簇时完成计算）。④从多个（非单一）母簇中选降低量最大的那一个分裂。⑤被分裂的母簇被内部已经二分割聚类生成的子簇（新生成的簇）替代，子簇变为新簇。⑥重新循环。⑦开始循环的第一步是新簇变母簇：新簇内部二分割聚类，共得到两簇母子分支。⑧每次新的循环的标志事件是：新进两簇母簇入选待分裂的母簇集，可以区分出每次循环的不同候选集和不同的分裂结果。

改进的二分聚类 选用的优化目标：母簇分裂的误差平方和降低量。

优化目标选择了母簇的母子分支的误差平方和降低量，而不是每个母簇分裂方案的总误差。类似决策树做分类的信息增益 Gain，不同的是非监督方式，其实大道相通。簇内的误差平方和越小，样本相互越靠近，越纯洁，越容易划分到一类中来。误差平方和降低量越大，说明得到簇的过程（到达肘部 elbow）越快，这一步迈得越大。 $SST=SSR+SSE$  总离差平方和=簇间（组间）平方和（反映偏差）+簇内（组内）残差平方和（反映方差）取  $X=\varepsilon^2$  即误差  $\varepsilon$  的估计量，根据方差性质  $E(X^2)=E(X)^2+D(X)$ ，可以得知系统偏差和方差的影响。这两个理论为选取误差平方和为聚类分析指标提供了尺度上的把握。极

端情况下：单一簇，误差平方和最大，为总离差；单样本簇，每个簇只有一个数据，误差平方和最小，为零。实际聚类情况：误差平方和达到应用要求的某一控制水平。误差平方和作为聚类的指示指标，是比较方便可行的。优化目标选择了母簇的母子分支的误差平方和降低量，而不是每个母簇分裂方案的总的误差平方和。这是为了程序选拔母簇有更加方便的针对指标，不需要关注总误差，只要具体到谁。母簇新生成时都通过二分聚类做了母子分支，并计算了母簇内的误差平方和降低量。此番改进不像标准的二分聚类程序，原有缺点是：每增加一个簇，要把各个母簇重复一次二分聚类，相当于重算一遍误差平方和降低量的工作量，而改进的程序仅对新生的母簇做二分聚类和误差统计。改进程序的存储空间保留了当前各本簇中心点、全体样本对各簇的从属关系、统计的本簇的误差项。增加了子簇中心点、各簇内样本对其各子簇的从属关系、子簇内的统计误差项。时空开销的权衡比较：空间最明显的增加就是子簇的从属关系，体量相当于全体样本对各簇的从属关系，增加了一倍的空间开销，但压缩了(K-2)的重复二分聚类和误差统计（最后一步 K 个聚类）的时间开销。

改进的适用情况：越多的聚类个数，越需要压缩重复二分聚类和误差统计的时间开销，尽管二分聚类每个母簇的样本数也在减小。改进的误差平方和降低量优化方法更加适合。

在阐述完上述原理之后，我们随后开始进行改进算法的测试，首先我们用 python 实现原本的如图 4.1.1 Kmeans 算法的代码,之后我们再实现改进了的如图 4.1.2 BiKmeans 的算法：

```
def kMeans(dataSet, k, maxIter = 5):  
    """  
    K-Means  
  
    Args:  
        dataSet: 数据集  
        k: 聚类数  
  
    Returns:  
        centroids: 聚类中心  
        clusterAssment: 点分配结果  
    """  
    # 随机初始化聚类中心  
    centroids = randCent(dataSet, k)  
    m, n = np.shape(dataSet)  
    # 点分配结果： 第一列指明样本所在的簇，第二列指明该样本到聚类中心的距离  
    clusterAssment = np.mat(np.zeros((m, 2)))#按：开始，各子簇簇号都是0号  
    # 标识聚类中心是否仍在改变  
    clusterChanged = True  
    # 直至聚类中心不再变化  
    #按：补充，发现空簇  
    kong=k*[0]#各簇 计数，初始值0  
    iterCount = 0
```

图 4.1.1 Kmeans 进核心代码



```

#提出的改进二分聚类

def biKmeans(dataSet, k): #按：改进二分聚类，不需要每次新增一簇就对全部的旧簇反复做二分聚类。
    """
    二分kmeans算法
    Args:
        dataSet: 数据集
        k: 聚类数
    Returns:
        centroids: 聚类中心
        clusterAssment: 点分配结果
    """
    m, n = np.shape(dataSet)
    # 起始时，只有一个簇，该簇的聚类中心为所有样本的平均位置
    # 本簇的初始化，包括中心，所属，误差三项
    centroid0 = np.mean(dataSet, axis=0).tolist()[0]
    # 设置一个列表保存当前的聚类中心
    currentCentroids = [centroid0]
    # 点分配结果：第一列指明样本所在的簇，第二列指明该样本到聚类中心的距离
    clusterAssment = np.mat(np.zeros((m, 2)))
    # 初始化点分配结果，默认将所有样本先分配到初始簇
    for j in range(m):
        clusterAssment[j, 1] = distEclud(dataSet[j, :], np.mat(currentCentroids))**2

```

图 4.1.2 bi- Kmeans 进核心代码

最后再进行聚类测试，通过测试同一组数据集的聚类效果，我们得到如下结果：

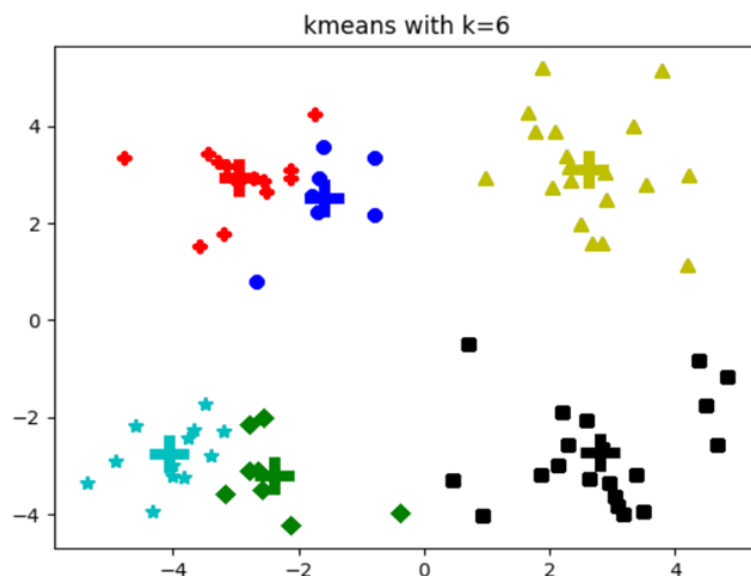


图 4.1.3 同一组数据 Kmeans 聚类结果

显然原本的 KMeans 在如图 4. 1. 3 中黑色簇聚类效果不是太好，红色的簇与黑色簇边界也没能很好的区分开来，虽然完成了聚类的任务，但是就实际聚类结果得到的图来看，还差了点精确性。因此就像本段开头所描绘的 Kmeans 缺点那样，在实际应用中 Kmeans 能完成聚类，但其效率与准确性表现得不是太好，这就是我们给出改进了的 Kmeans，即 BiKmeans 的意义所在了，下面请看如图 4. 1. 4BiKmeans 聚类效果：



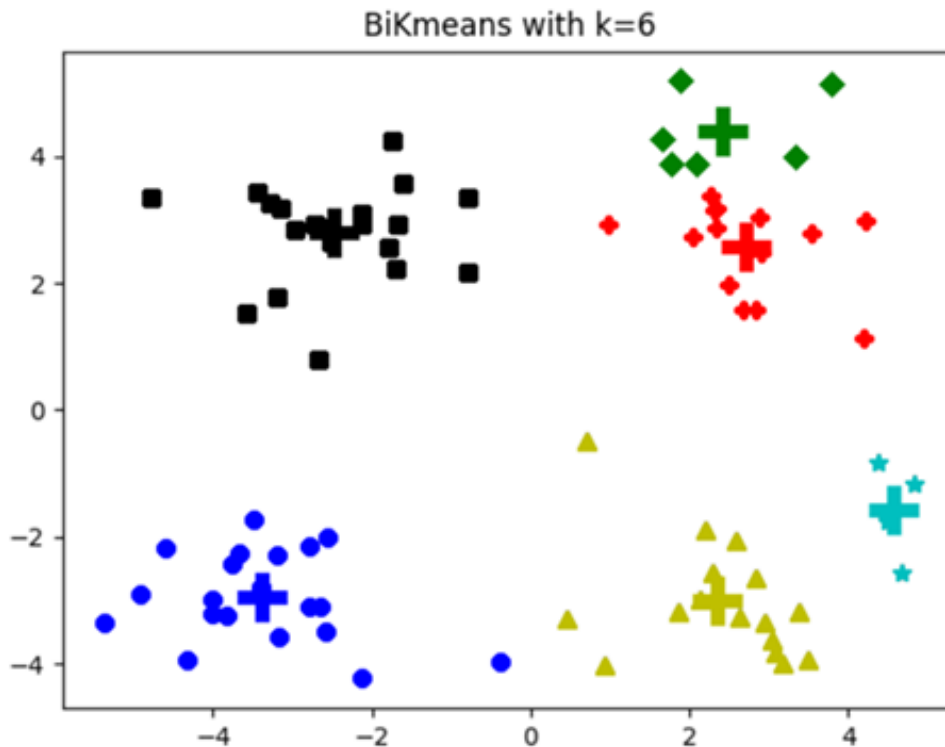


图 4.1.2 同一组数据 BiKmeans 聚类结果

显然 BiKmeans 比 Kmeans 聚类要准确地多。

#### 4.3 DBSCAN 算法的缺点

- (1) DBSCAN 对圆半径  $\epsilon$  和 MinPts 最小阈值的设置敏感。
- (2) 如果集群稀疏不同，集群效应也会大不相同，则很难使用该算法。
- (3) 如果数据样本集大，那么 DBSCAN 聚类的收敛所花费的时间就会变得长。

#### 4.4 DBSCAN 算法的改进

针对这些缺点我们提出了半自动 DBSCAN 以及自动 DBSCAN，正是因为 DBSCAN 对  $\epsilon$  和 MinPts 特别敏感，从而影响其效率与准确性，我们才能针对这个问题进行优化。半自动 DBSCAN 核心代码如图 4.2.1，自动 DBSCAN 聚类结果如图 4.2.2：

```

43
44
45 # 专家凭经验给出一批适合这批数据点分布特征的MinPtsProfessor参数值
46 def inputMINPTSPROFESSOR():...
47
48
49 # 从MINPTSPROFESSOR中循环取出数据，从而获得EPSMIN列表
60 def getEPSMINI(Dsort, MINPTSPROFESSOR):...
61
62
63 def getLESSEPSMINI(Dsort, EPSMIN, MINPTSPROFESSOR):...
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Professor Input 5 number as MinPts:

专家给出的MinPts参考值:

MINPTSPROFESSOR: [5, 2, 1, 3, 4, 1]

依据每个MinPts参考值而找到每一行中的最小的EPS值:

图 4.2.1 半自动 DBSCAN 核心代码

可以看到我们能够跟具实际情况去设置五个 MinPts, 根据我们以往的数据分析的经验来调整, 或者说优化 DBSCAN 的性能, 使得 DBSCAN 能够减少错误在半自动人为的干涉下从而提高准确性, 这个优化思路就是反向利用了该算法对参数敏感的特性, 人的大脑始终比机器的大脑思考问题复杂的多, 所以在经验认知上面占据一定的优势。显然, 在这种半自动的 DBSCAN 比没有优化的 DBSCAN 要好。

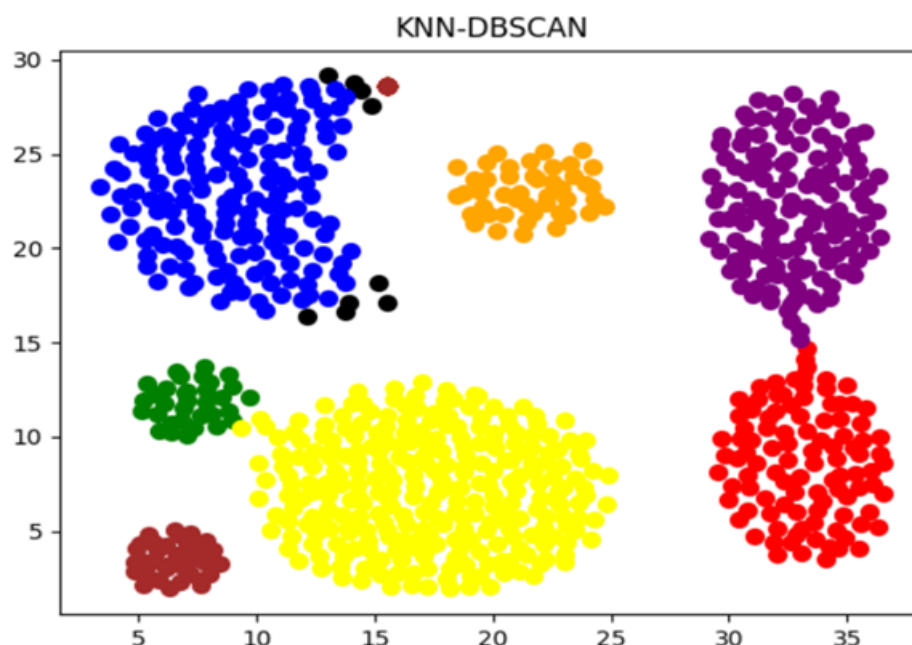


图 4.2.1 自动 DBSCAN 聚类结果

可以看到自动 DBSCAN 与半自动 DBSCAN 都完成了任务, 而且可以自动或半自动设置 DBSCAN 敏感的那两个参数, 从而找到最优的参数。

#### 4.5 Agglomerative 算法的缺点

- (1) 计算复杂度太高。
- (2) 奇异值也能产生很大影响。
- (3) 算法很可能聚类成链状。

#### 4.6 Agglomerative 算法的改进

BIRCH 算法使用树结构来创建集群。它通常被称为聚类特征树 (CF Tree)。这棵树的每个节点都由几个聚类特征 (CF) 组成。它是一种内存效率高的在线学习算法, 可作为 MiniBatchKMeans。它构造了一个树形数据结构, 其中簇中心从叶子中读取。这些可以是最终的聚类质心, 也可以作为另一个聚类算法的输入。此外 BIRCH 运行起来非常检查, 仅仅需要对所使用的数据扫描一次, 便可以把大型数据集快速处理完毕。该算法基于 CF (聚类特征) 树。此外, 该算法使用树结构摘要来创建集群。给定数据的树结构是由称为聚类特征树 (CF 树) 的 BIRCH 算法构建的。在 CF 树的上下文中, 该算法将数据压缩成 CF 节点集。那些具有多个子集群的节点可以称为 CF 子集群。这些 CF 子集群位于无终端 CF 节点中。CF 不仅是一种平衡度高的树, 而且为了进行的数据层次聚类, 它还会收集整理好聚类的特征。这避免了使用作为输入的整个数据的

需要。作为 CF 的数据点的树簇由三个数字（N、LS、SS）表示。N 代表子集群中的项目数，LS 代表数据点的向量和 SS 代表平方数据点的总和。BIRCH 算法主要遵循四个阶段。①将数据扫描到内存中。②压缩数据（调整数据大小）。③全局聚类。④精炼集群。

在这四个阶段中，其中两个（调整数据大小和优化集群）是可选的。当需要更清晰时，它们会出现在这个过程中。但是扫描数据就像将数据加载到模型中一样。加载数据后，算法会扫描整个数据并将它们拟合到 CF 树中。在压缩过程中，它会重置数据并调整其大小，以便更好地适应 CF 树。在全局聚类中，它使用现有的聚类算法发送 CF 树进行聚类。最后，精炼解决了 CF 树的问题，即相同值的点被分配给不同的叶节点。

Scikit Learn 在集群类包下提供了直接实现 BIRCH 的模块。我们需要根据要求为参数提供值。BIRCH 算法中有三个参数。①CF 树中叶节点的子集群中要考虑的最大数据样本数。②Branching\_factor - 它是用于指定可以在节点中创建的 CF 子集群数量的因子。③N\_clusters - 集群数量。再介绍完毕这些之后，我们使用 python 实现了如图 4.3.1 的 BIRCH 算法，最后的聚类效果如图 4.3.2。

```
from sklearn.datasets import make_blobs
from sklearn.cluster import Birch
import matplotlib.pyplot as plt

# 生成样本点
centers = [[1, 1], [-1, -1], [1, -1]]
X, labels = make_blobs(n_samples=750, centers=centers,
                       cluster_std=0.4, random_state=0)

clustering = Birch(n_clusters=3).fit(X)
plt.figure(figsize=(10, 8))
plt.scatter(X[:, 0], X[:, 1], c=clustering.labels_, cmap='prism')
plt.title('Birch')
plt.show()
```

图 4.3.1 sk-learn 中 Birch 代码

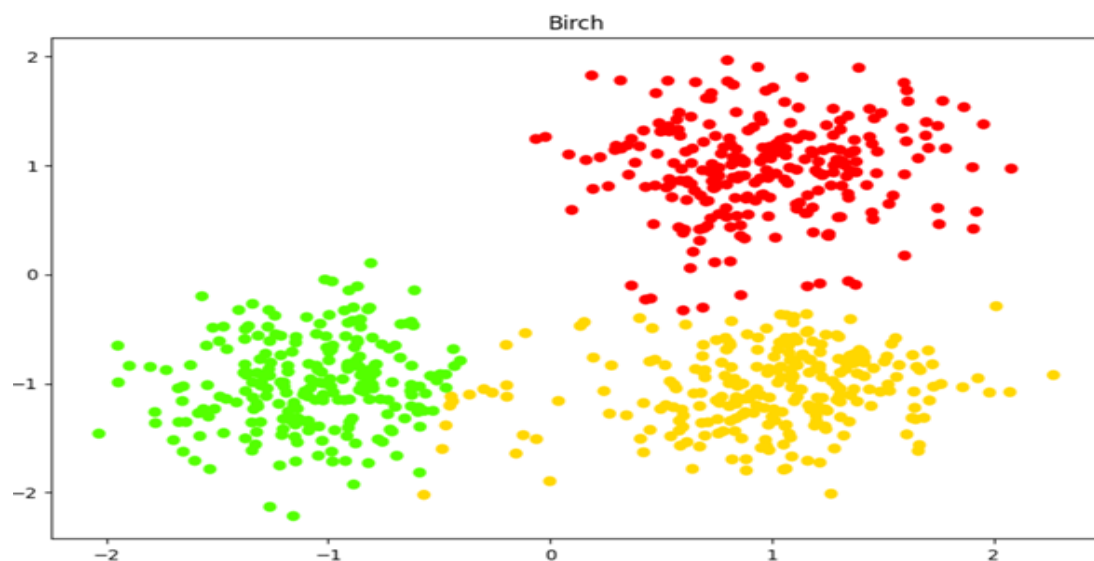


图 4.3.2 Birch 聚合结果

## 5 总结

本文介绍了常用的聚类算法概念以及代码实现,并分析了聚类算法的主要优势和存在缺陷,也应用 IsolationForest 这一聚类算法解决了 2021 年五一数学建模 C 题的问题。虽然聚类算法经有二十年的历史,但它还在不断发展并保持与时俱进。从划分聚类,再到层次聚类,再到密度聚类。尽管聚类算法还存在着许多不足之处,但随着专业人士的改进,聚类算法再不断完善与增强,以便人们用于数据挖掘、数据分析。我们有理由相信这一技术在未来仍然会带给我们许多便利。

## 参考文献

- [1]张洋. 基于百度搜索指数的 CPI 短期预测研究[D].天津财经大学,2016.
- [2]孙思. 利用遗传思想进行数据划分的 DBSCAN 算法研究[D].重庆大学,2005.
- [3]黄毅磊. DBSCAN 算法及在城市网格化管理中的应用[D].上海交通大学,2010.
- [4]于珊珊. 基于模糊分类的遥感影像变化监测研究[D].辽宁师范大学,2018.
- [5]张荣昌. 基于数据挖掘的用电数据异常的分析与研究[D].北京交通大学,2017.
- [6]虞倩倩. 基于数据划分的 DBSCAN 算法研究[D].江南大学,2013.
- [7]田路强. 基于 DBSCAN 的分布式聚类及增量聚类的应用研究[D].北京工业大学,2016.
- [8]陈玉霞.基于 SARIMA 模型的贵州省季度 GDP 预测[J].经营与管理,2021(08):170-175.
- [9]任景华.利用优化的 DBSCAN 算法进行文献著者人名消歧[J].图书馆理论与实践,2014(12):61-65.
- [10]樊仲欣.基于数据流的聚类趋势分析算法[J].计算机应用,2020,40(08):2248-2254.
- [11]伍艺. 面向大数据集的递增聚类方法研究[D].北京理工大学,2015.
- [12]张洋. 基于百度搜索指数的 CPI 短期预测研究[D].天津财经大学,2016.
- [13]孙思. 利用遗传思想进行数据划分的 DBSCAN 算法研究[D].重庆大学,2005.
- [14]付希. 基于蚁群算法的聚类分析在学生成绩评价中的应用研究[D].西南交通大学,2013.
- [15]iFlyA. Python 机器学习笔记：异常点检测算法——LOF (Local Outlier Factor) . [2020-12-03].  
<https://blog.csdn.net/iFlyAI/article/details/110531125>.
- [16]战争热诚. Python 机器学习笔记：异常点检测算法——Isolation Forest . [2019-04-13].  
<https://www.cnblogs.com/wj-1314/p/10461816.html>.
- [17]石楚臣. 基于聚类算法和 Spark 框架的电量与线损分析研究 [D]. 南京师范大学,2020.DOI:10.27245/d.cnki.gnjsu.2020.000561.
- [18]denghe1122 . 聚类算法 ( 一 ) —— k-means 算法 以及 其 改进 算法 . [2018-09-17].  
<https://blog.csdn.net/dengheCSDN/article/details/82744181>.
- [19]石楚臣. 基于聚类算法和 Spark 框架的电量与线损分析研究 [D]. 南京师范大学,2020.DOI:10.27245/d.cnki.gnjsu.2020.000561.

## 致谢

最后一个季节总是阳光明媚。我们六月份毕业了。每年凤凰花开的时候，都是一个学生去另一个“世界”的旅程，结束了校园的生活，多年的生活，不过回首往事真的很甜蜜，这些也会随着我们的毕业而留下美好的回忆，那将永远是怀旧的，独一无二的！在论文的最后，请允许我对帮助我完成论文、学习和生活的我亲爱的家人和导师表示衷心的感谢！感谢我的导师刘淑影。她的思想和话语充满了诗意、哲理和魔力。他们在我的脑海里唤醒了多少美妙的波浪！老师，你是指引我们前进的灯塔。你在我们里面。在繁重的科研和教学任务中，刘老师主动照顾我的科研和学习。帮助我一起解决论文的不足，从选题、撰写开篇报告、信息检索、结构改进等方面进行详细的指导，使我能够顺利完成论文，同时也谢谢我的家人们一直伴随在我身旁。我才能够全身心地投入到学习和工作中，并成功地完成学业。