

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期：**

**计算机科学与技术学院**

**目录**

[实验四 kmeans算法及其实现 1](#_Toc58793873)

[**4.1实验目的** 1](#_Toc58793874)

[**4.2 实验内容** 1](#_Toc58793875)

[**4.3 实验过程** 2](#_Toc58793876)

[4.3.1 编程思路 2](#_Toc58793877)

[4.3.2 遇到的问题及解决方式 2](#_Toc58793878)

[4.3.3 实验测试与结果分析 2](#_Toc58793879)

[**4.4 实验总结** 2](#_Toc58793880)

# 实验四 kmeans算法及其实现

## **4.1实验目的**

1、加深对聚类算法的理解,进一步认识聚类算法的实现；

2、分析kmeans流程,探究聚类算法原理；

3、掌握kmeans算法核心要点；

4、将kmeans算法运用于实际，并掌握其度量好坏方式。

## **4.2 实验内容**

提供葡萄酒识别数据集，数据集已经被归一化。同学可以思考数据集为什么被归一化，如果没有被归一化，实验结果是怎么样的，以及为什么这样。

同时葡萄酒数据集中已经按照类别给出了1、2、3种葡萄酒数据，在cvs文件中的第一列标注了出来，大家可以将聚类好的数据与标的数据做对比。

编写kmeans算法，算法的输入是葡萄酒数据集，葡萄酒数据集一共13维数据，代表着葡萄酒的13维特征，请在欧式距离下对葡萄酒的所有数据进行聚类，聚类的数量K值为3。

在本次实验中，最终评价kmean算法的精准度有两种，第一是葡萄酒数据集已经给出的三个聚类，和自己运行的三个聚类做准确度判断。第二个是计算所有数据点到各自质心距离的平方和。请各位同学在实验中计算出这两个值。

实验进阶部分：在聚类之后，任选两个维度，以三种不同的颜色对自己聚类的结果进行标注，最终以二维平面中点图的形式来展示三个质心和所有的样本点。效果展示图可如图1.1所示。



图4.1 葡萄酒数据集在黄酮和总酚维度下聚类图像（SSE为距离平方和，Acc为准确率）

## **4.3 实验过程**

### 4.3.1 编程思路

本实验需要运用到k-means算法，根据归一化.csv文件中的十三维数据，来将这些数据点在二维平面图中聚成3个簇中，并且以点图的形式表示出来。

本次实验涉及到几个比较重要的概念：其一是簇，簇是指具有相似特征的数据对象的集合，在此次实验中我们将把这些点分成三个簇；其二是簇中心，这个点是簇中所有数据对象的均值向量，可以一定程度上表示簇的特征，在本次实验中其未簇内所有数据点的平均值；其三是距离，本次实验中我们使用欧式距离来衡量数据对象之间的相似度；其四是损失函数，我们使用簇内平方误差和(SSE)来判断，这个值是来衡量聚类的效果，其计算公式如下：

其中，K是簇的个数，是第i个簇中的所有数据点的集合，是第i个簇的中心点（即簇中所有数据点的平均值向量），SSE的值越小说明聚类生成的簇效果越好。

首先讲一下k-means算法的大概思路：我们首先会提前确定几个常数：k、tolerance、max\_iter。它们分别是簇类的个数、差值和迭代次数，本次实验没有给定迭代次数和差值，我们假设其为300和0.0001方便后续迭代时处理。算法开始时，会随机选择3个点作为第一次迭代的初始中心点。对于数据集中的每个数据点，我们去计算这个点和3个聚类中心点之间的距离，然后根据距离大小将其划分到距离最近的那个聚类中心所在的簇。这样分好所有的点后，我们对于每个簇中的点再去重新计算簇中所有点的平均值，得到了下一次迭代的新的中心点。重复上述的几个步骤直到到达迭代次数或者满足差值为止，我们会得到3个簇，到此k-means算法结束，我们只需要再去分析精准度（accuracy）和SSE即可。

我们首先要读取“归一化数据.csv”文件，文件内容第一行是已经分好的类别（1，2，3），后面的十三栏是葡萄酒的十三维数据，部分内容如下图所示：

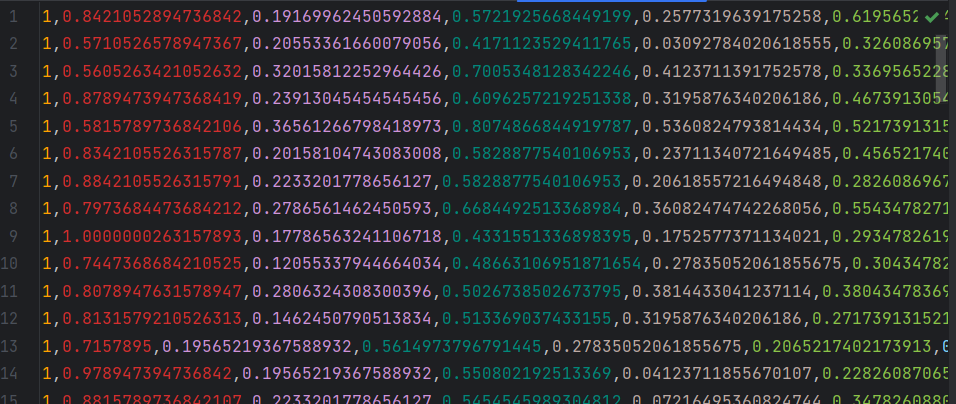


图4.2 归一化数据.csv部分内容

我们通过pandas库中的read\_csv函数来读取文件内容，并将csv文件中除了第一列的数据以外的数据转换为Numpy类型数据。在后续主函数中，利用list函数将其类型改变。相关代码如下图所示：

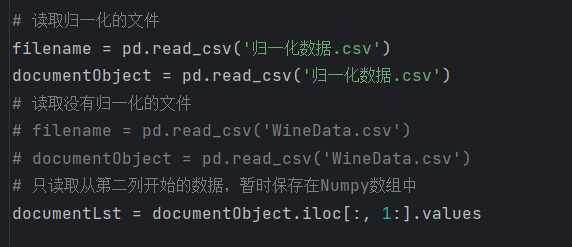


图4.3 将文件中的除第一列数据外，转化成Numpy数组

接着我们定义一个python的类，名为“k-means”，并且在“\_\_init\_\_”方法中将提前设定好的几个参数定义出来，方便后续的迭代处理，相关代码如下图所示：

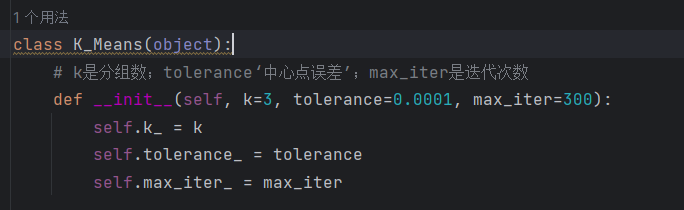


图4.4 k-means类中定义的三个参数

接着我们在fit方法中实现迭代步骤，首先要初始化3个中心点，以开始第一次的迭代。这一步使用random函数来实现随机初始化3个中心点，关键代码如下图所示：

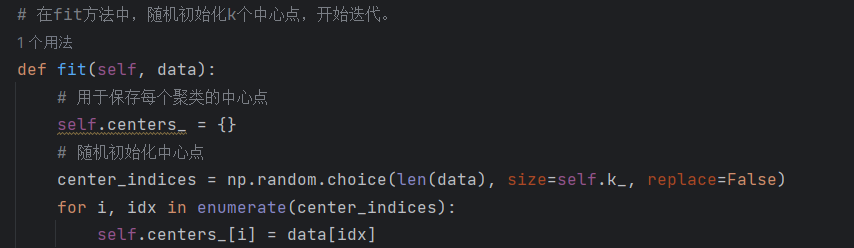
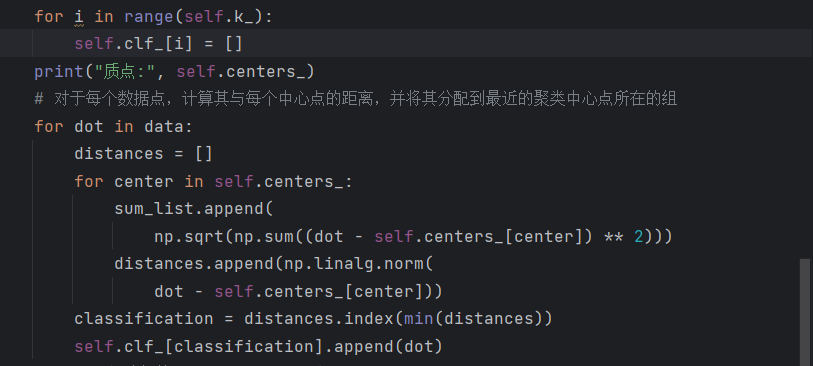


图4.5 随机初始化3个中心点

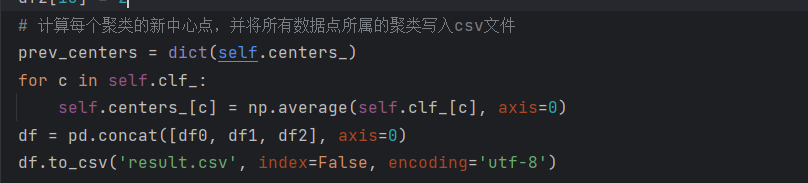
确定好3个中心点后，先初始化分好三个簇存入self.clf中，在遍历中对于每个点去计算到这三个中心点的距离，并且根据距离的大小将其分配到不同的中心点中去。在这一步利用两个for循环，先遍历所有点然后对每个点遍历所有的中心点，计算距离后找到最小距离的中心点，将这个点存入对应的簇。相关代码如下图所示：



**计算到中心点的距离**

图4.6 计算每个点到中心点的距离

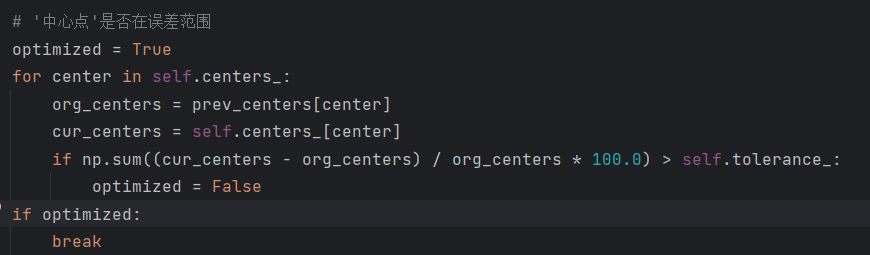
根据每个簇中的数据重新计算簇的中心点，方便下一次迭代的处理，并且判断两者误差是否符合tolerance的值，相关代码如下图所示：



**计算新的中心点**

图4.7 计算每个簇中的平均值，找寻下一个中心点

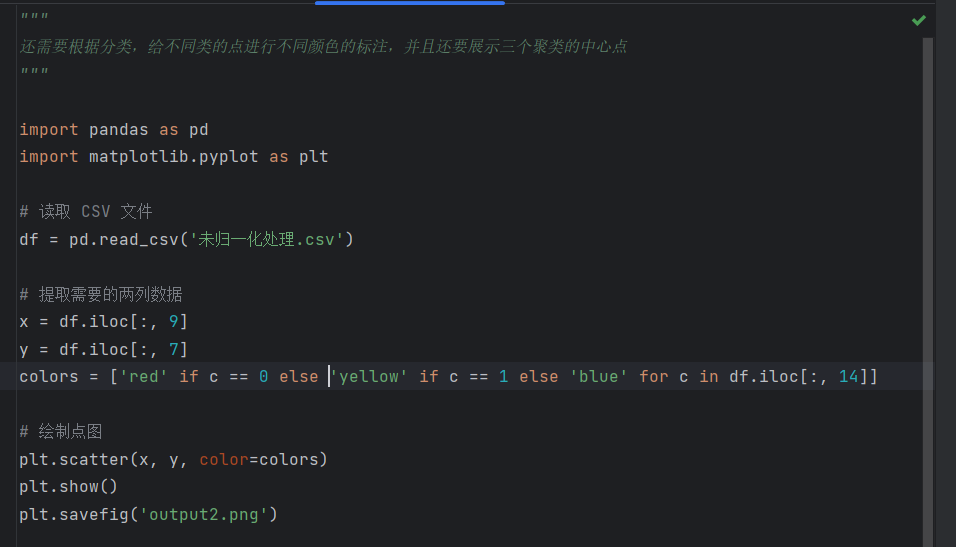
在算法的每次迭代中，会计算每个聚类的新中心点，然后比较新旧中心点之间的误差。当所有聚类的中心点误差都小于给定的tolerance值时，就认为聚类结果已经优化，此时变量optimized会被置为True，迭代结束。如果某个聚类的中心点误差超过tolerance值，则将optimized置为False，继续迭代。相关代码如下图所示：



**判断是否在误差范围内**

图4.8 判断是否在误差范围内

我们需要将分配好的点，以三个颜色来区分不同簇的点，对于k-means算法处理得到的CSV文件进行数据读取和提取，然后对其中两列数据进行可视化绘制。首先，代码利用 Pandas 库的 read\_csv() 方法读取名为 'result.csv' 的 csv文件，使用 iloc 方法提取出文件中的第9列和第7列的数据分别赋值给 x 和 y 两个变量。接下来，根据 csv 文件中第14列的分类信息，将不同簇的点分别标记为红色、黄色和蓝色，在二维点图展示出来。最后通过使用 plt.show() 方法来显示绘制好的点图，并使用 plt.savefig() 方法将图像保存为名为 'output.png' 的文件。相关代码如下图所示：



**根据两列数据绘制点图**

图4.9 根据处理后的csv文件绘制点图

### 4.3.2 遇到的问题及解决方式

**问题一：如何实现随机生成3个中心点？**

在我们从csv文件获取所有点的数据之后，我们了解到点的个数为178，所以我们只需要在1~178中随机生成三个整数作为点的列表索引即可得到三个随机的中心点。在python中，random库给我们提供了random.sample的函数，我们需要确定范围使用range(1,178)，相关代码如下图所示：

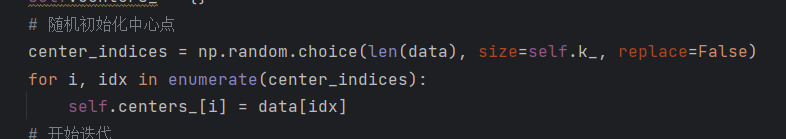
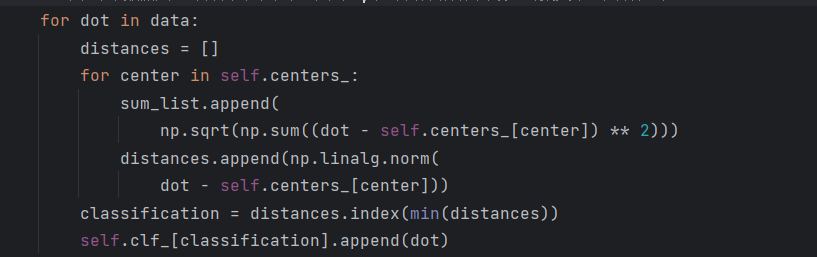


图4.10 随机化三个簇的中心点

**问题二：如何计算SSE，也即如何留存每个点到中心点的距离？**

由于每次迭代中，我们都提前求出过所选中心点的坐标，所以我们只需要遍历一下存入了所有点数据的data列表里，采用欧式距离计算到中心点的距离。不仅要将同一个点的三个距离存入一个列表distances里，还要将这些距离存入一个单独的列表里，方便后续处理。相关代码如下图所示：



**根据距离大小将其分给不同的簇**

图4.11 计算SSE

### 4.3.3 实验测试与结果分析

**实验测试：**

本次实验需要将组类平方和SSE和精确度输出，将所有代码按照要求运行，所得的最后结果如下图所示。值得注意的是由于第一遍选取点的随机，最后的SSE和准确度都不会相同，这次结果尽量选取了SSE较小的一次结果，实际上会随选取第一轮中心点的不同而改变，结果如下图所示：

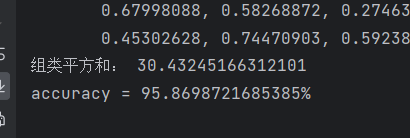


图4.12 实验结果

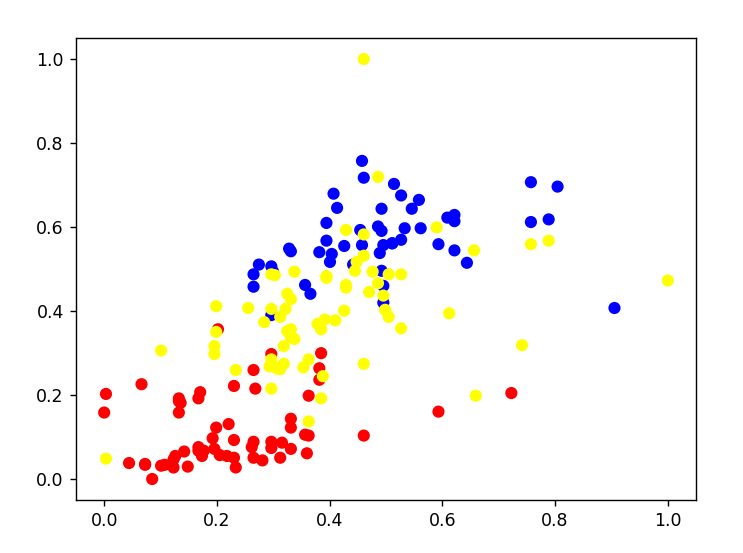
 在k-means算法结束后，使用create\_map.py将分类的结果根据所选取的两列数据来画出二维点图，最后得到的结果如图所示：

图4.13 自己编写的算法运行后的二维点图

下面是直接使用库中的k-means函数直接产生的二维点图：

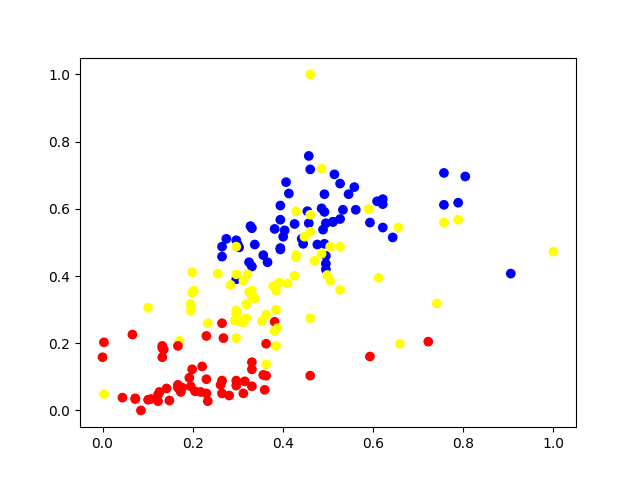


图4.14 直接使用库函数产生的聚类结果二维点图表示

下面是直接使用没有归一化的数据直接产生的二维点图：

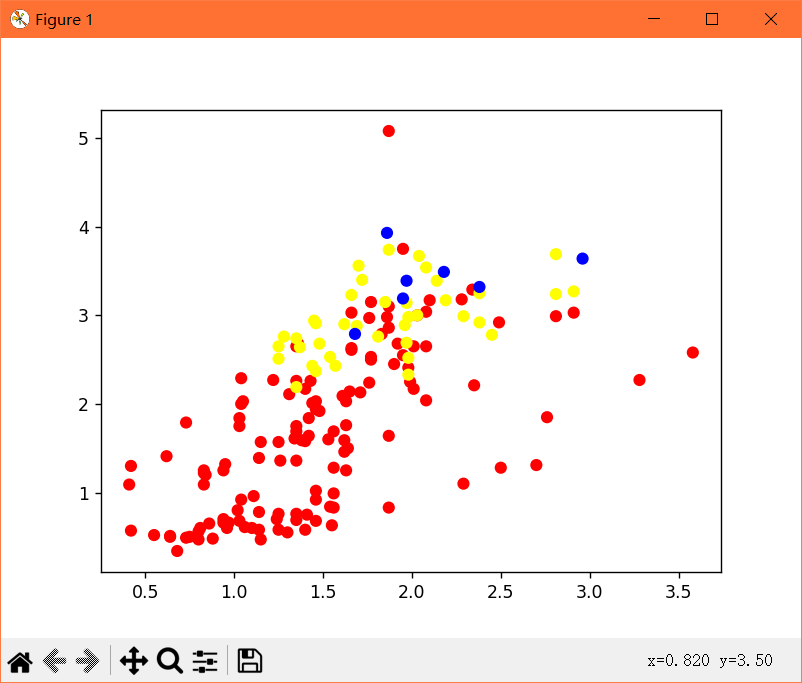


图4.15 直接使用未归一化聚类结果二维点图表示

**结果分析：**当以使用库函数的二维点图结果作为衡量标准时，我们可以发现本次实验的聚类效果还不错，只有少数点的聚类出现了问题。我认为这可能是由于随机选取的点或者预设的常量不够准确所致，需要进行更多次的迭代或降低容忍度来改进。

对于未归一化的数据，我们可以明显看出聚类效果非常差。这主要是因为数据需要进行归一化处理。我查询了一些资料后了解到，以本次葡萄酒数据集为例，数据集中的每个样本都包含了13个特征，包括ph值、酒精含量、含糖量等等。这些特征的单位不同，甚至单位的量级也各不相同。如果不对这些特征进行归一化处理，K-Means聚类算法中的某些特征将比其他特征更具有权重，从而影响聚类结果。例如，某些维度的数据量级在几十到几百，而有些数据只有零点几的量级，这会严重影响聚类算法的效果。

## **4.4 实验总结**

本次实验我首先认识到了归一化处理的重要性。通过归一化处理我们可以将数据的特征缩放到我们所需要的一个比例和范围内，使得所有特征对聚类结果的影响差距尽量减小甚至消除。假设我们不对这些数据进行归一化，可能让聚类结果不准确。

这次实验主要学习并实践了k-means算法，可以将数据集划分为多个簇，每个簇内部的数据相似度较高，而簇之间的数据相似度较低。我们选择使用葡萄酒数据集进行聚类分析，该数据集包含了 多个特征，用来描述葡萄酒的不同属性。有些特征的数据范围不同，因此需要对数据进行归一化处理，以确保所有特征的数据范围都在 0-1 之间，避免某些特征对聚类结果产生过大的影响。为了用精确数据来衡量聚类结果，我们可以通过SSE和准确度来判断，其中我们在迭代中**尽量使得簇内距离越来越小，而簇与簇之间的距离越来越大。**在分好类后，我们通过和数据集给出的类别作比较，我们可以计算精确度。并且我也通过使用matplotlib.pyplot库中的相关函数来绘制二维点图。

最后，本次实验让我了解聚类算法的原理和实现方法，并通过实际操作来了解算法的应用场景和需要注意的地方。同时，也让我们掌握了一些数据分析和可视化的基本技能。