机器学习算法及其实现

作业要求

在上次提供的数据集上分别实现对应的算法（当然也可以在自己找的数据集上）：

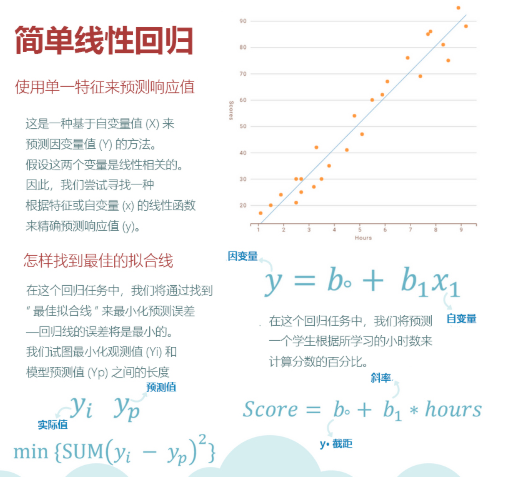
1、回归（Air quality dataset）：逻辑回归；

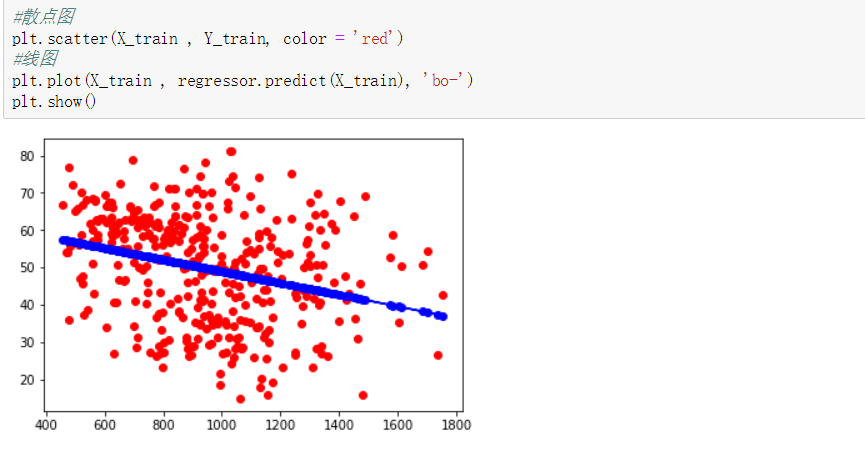
2、分类（BLE&RSSI dataset）：SVM、决策树、随机森林；

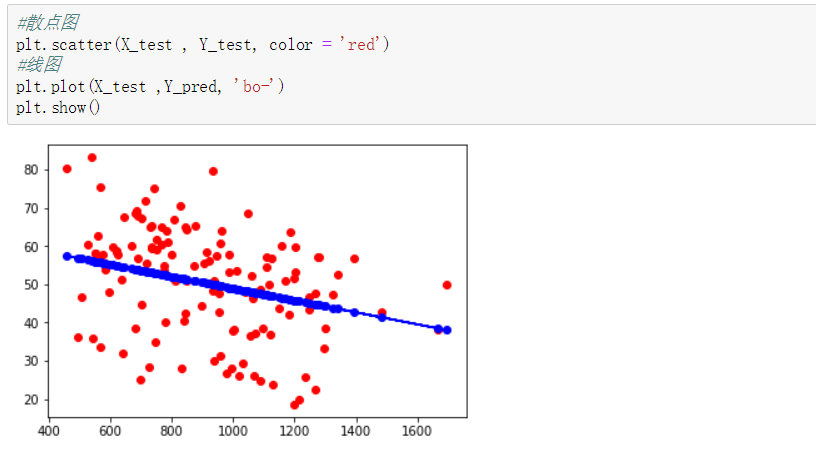
3、聚类（BLE&RSSI dataset）：DBScan、kmeans、GMM、层次聚类算法

其中聚类算法要求以t-SNE实现结果可视化

* 线性逻辑回归



* 运行结果：  
  训练集：  
    
  测试集：

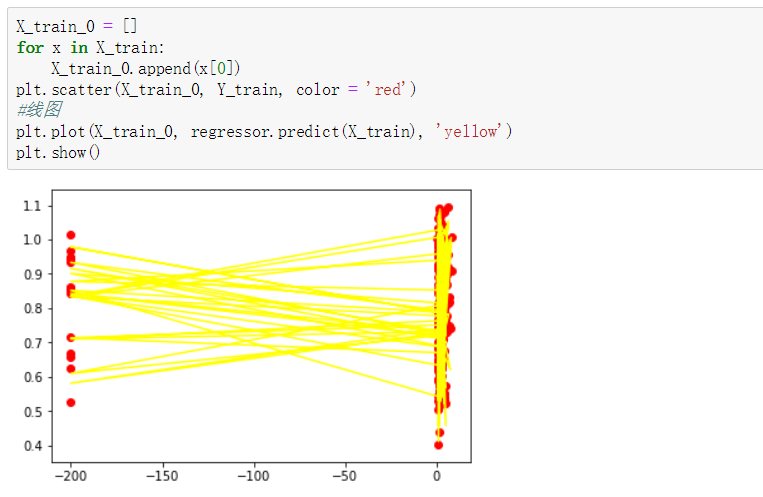


* 多重线性回归

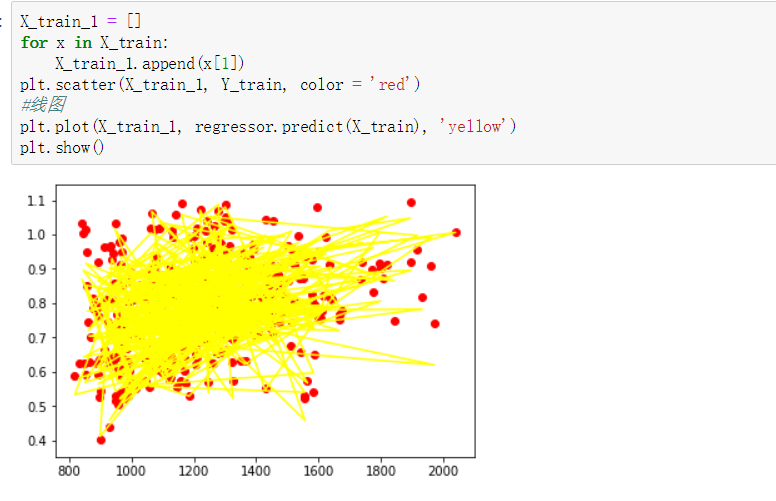


运行结果：

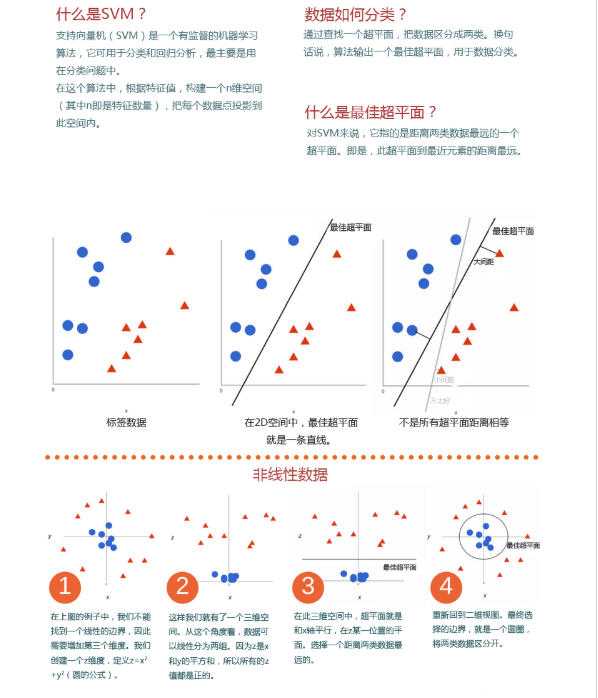
对多元变量的第一个变量的测试结果：



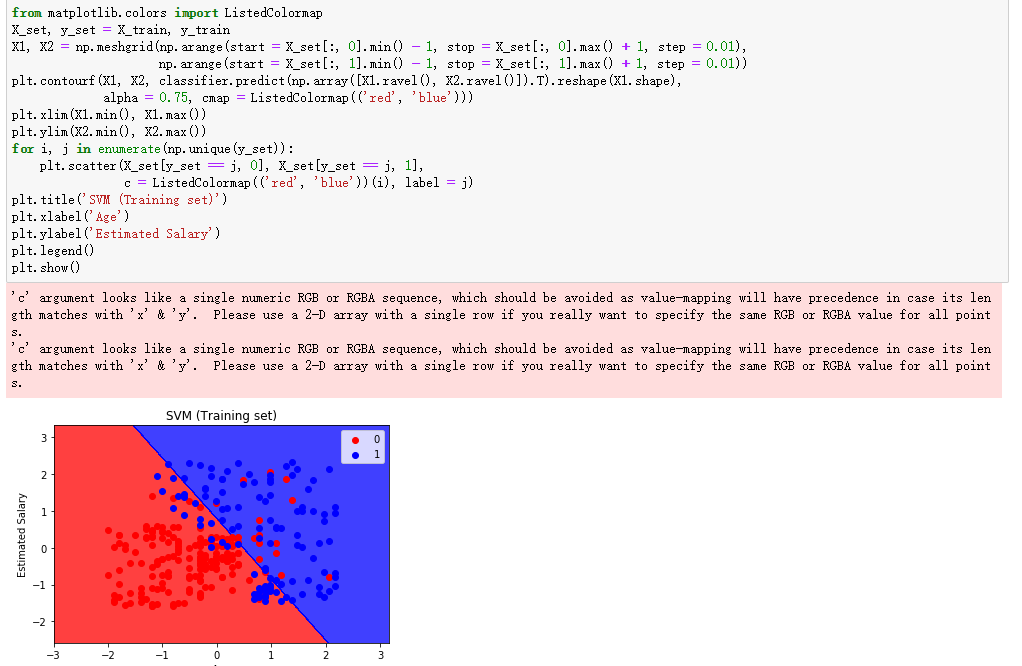
对多元变量的第二个变量的测试结果：



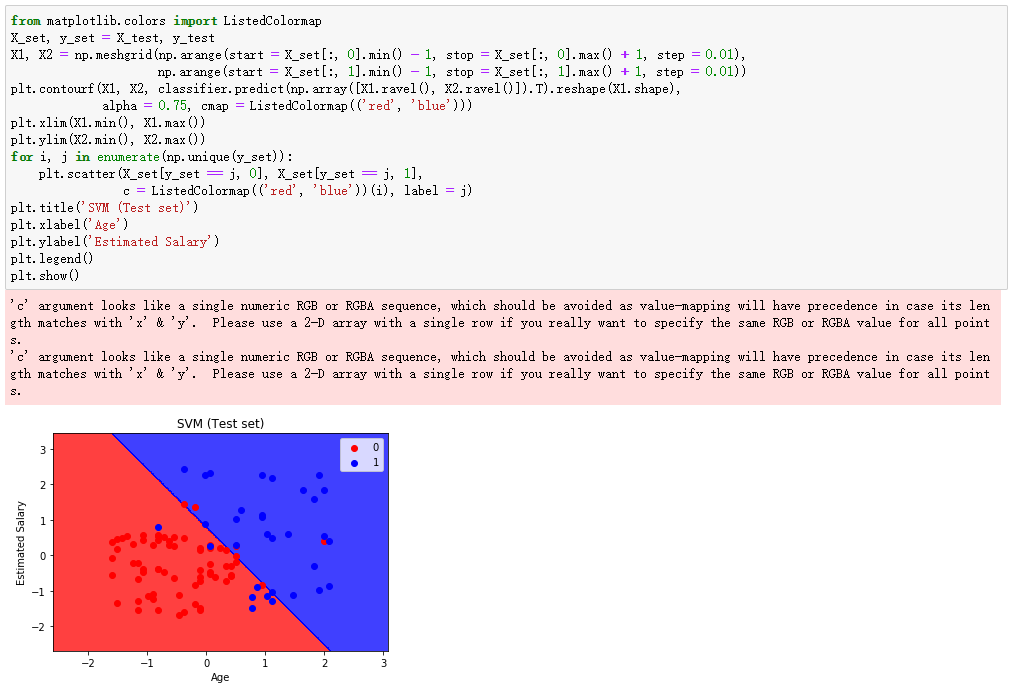
* SVM

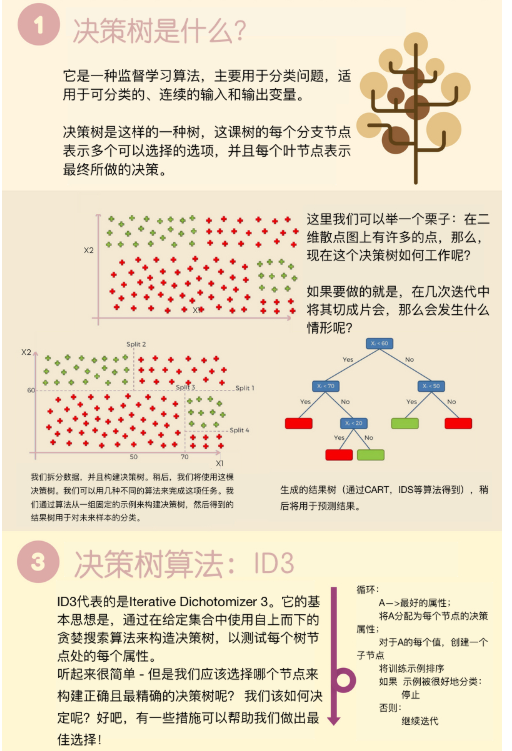
  
数据集测试结果：

训练集

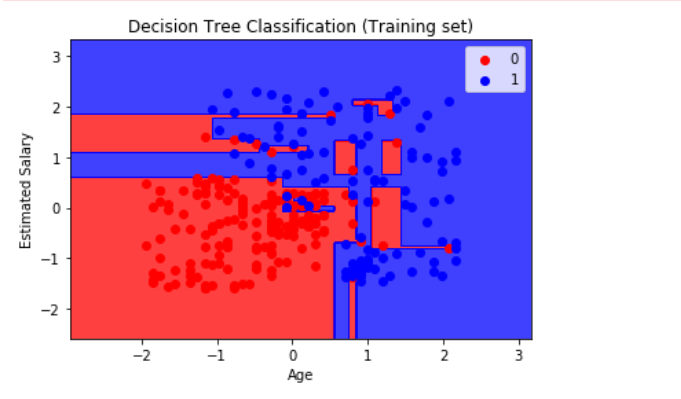


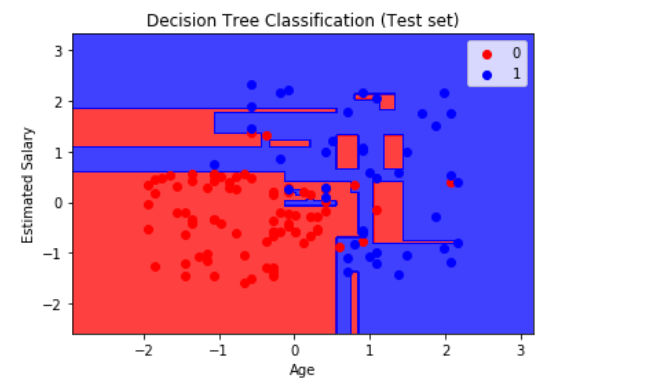
测试集

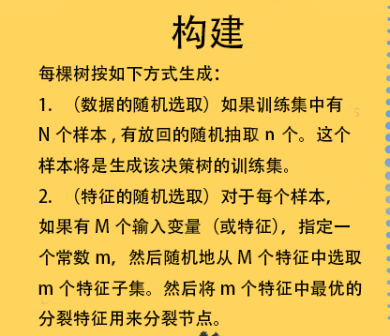


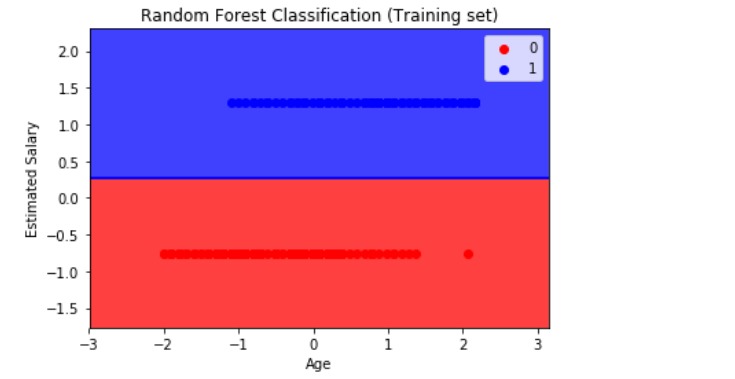
* Decision\_Trees  
  

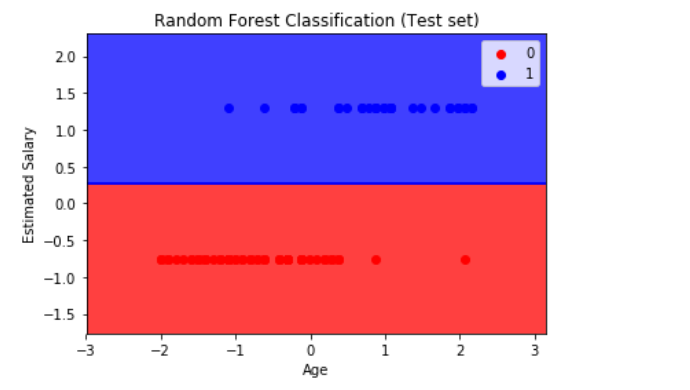
运行结果：





* Random\_Forests  
  

实验结果：



* DBSCan

**DBSCAN算法的聚类过程**

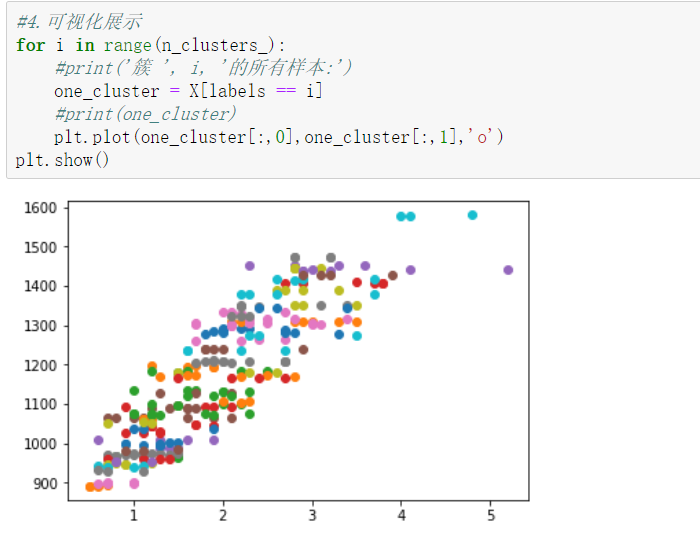
DBSCAN算法基于一个事实：**一个聚类可以由其中的任何核心对象唯一确定**。等价可以表述为：任一满足核心对象条件的数据对象p，数据库D中所有从p**密度可达**的数据对象o所组成的集合构成了一个完整的聚类C，且p属于C。

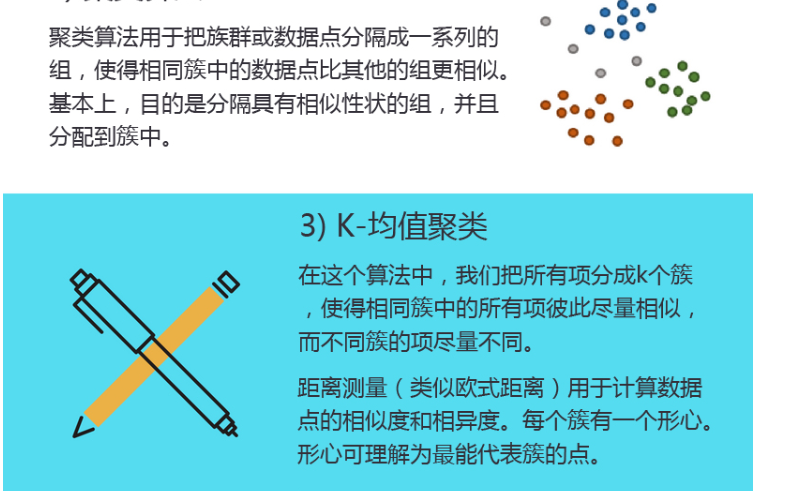
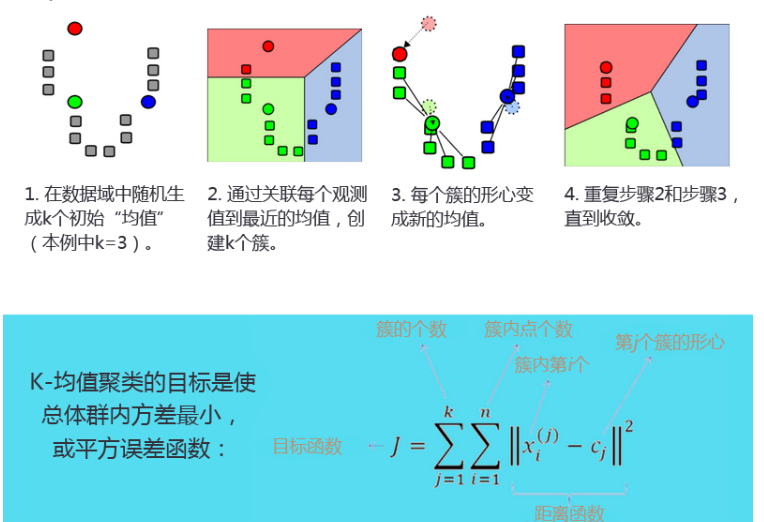
算法的具体聚类过程如下：

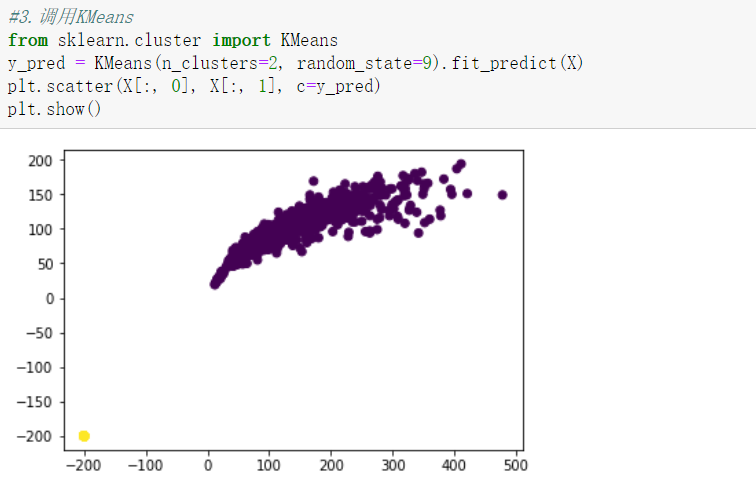
**扫描整个数据集，找到任意一个核心点，对该核心点进行扩充。扩充的方法是寻找从该核心点出发的所有密度相连的数据点（注意是密度相连）。遍历该核心点的**[clip_image006[4]](https://images0.cnblogs.com/blog/407700/201307/01135230-8e18685d32b346c187cbfd1ffbdce02d.gif)**邻域内的所有核心点（因为边界点是无法扩充的），寻找与这些数据点密度相连的点，直到没有可以扩充的数据点为止。最后聚类成的簇的边界节点都是非核心数据点。之后就是重新扫描数据集（不包括之前寻找到的簇中的任何数据点），寻找没有被聚类的核心点，再重复上面的步骤，对该核心点进行扩充直到数据集中没有新的核心点为止。数据集中没有包含在任何簇中的数据点就构成异常点。**



实验结果：



* Kmeans聚类  
    
  

实验结果：  


* GMM  
  **.高斯混合模型**

      K个GSM混合成一个GMM，每个GSM称为GMM的一个component，也就是分为K个类，与K-means一样，K的取值需要事先确定，具体的形式化定义如下：

https://images0.cnblogs.com/blog/512363/201304/24162207-ac46477f8817483481aa593fda26929c.gif

      其中，https://images0.cnblogs.com/blog/512363/201304/24162227-39898c96145e4ea38db2086d0815ec77.gif是样本集合中k类被选中的概率：https://images0.cnblogs.com/blog/512363/201304/24162553-bb8e9dc6fdfa40cc9136ca37d550399d.gif，其中z=k指的是样本属于k类，那么https://images0.cnblogs.com/blog/512363/201304/24162617-ded5de9c60c24295bac4c0cf7068d865.gif可以表示为https://images0.cnblogs.com/blog/512363/201304/24162632-98fa6c26905c4eb6b025b45b0dba852e.gif,很显然https://images0.cnblogs.com/blog/512363/201304/24162645-fdfb6fe72b5e49419757048225816e36.gif，y是观测数据。

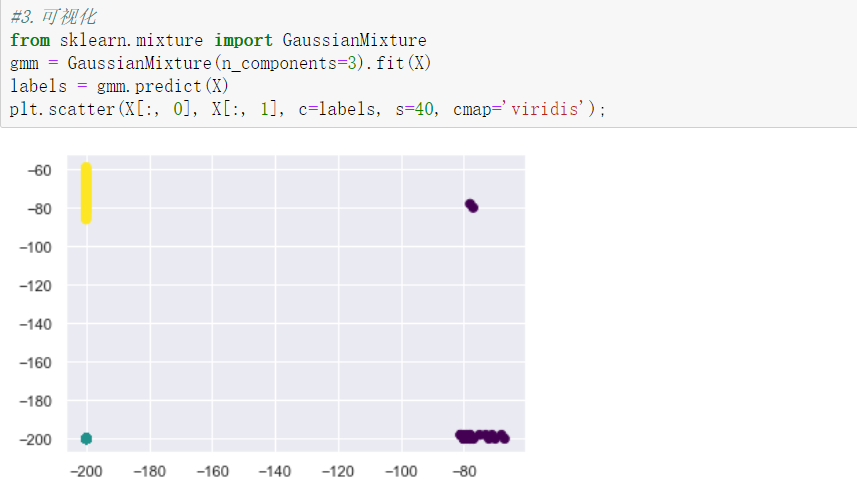
       这里如果我们事先知道每个样本的分类情况，那么求解GMM的参数非常直观，如下表示：

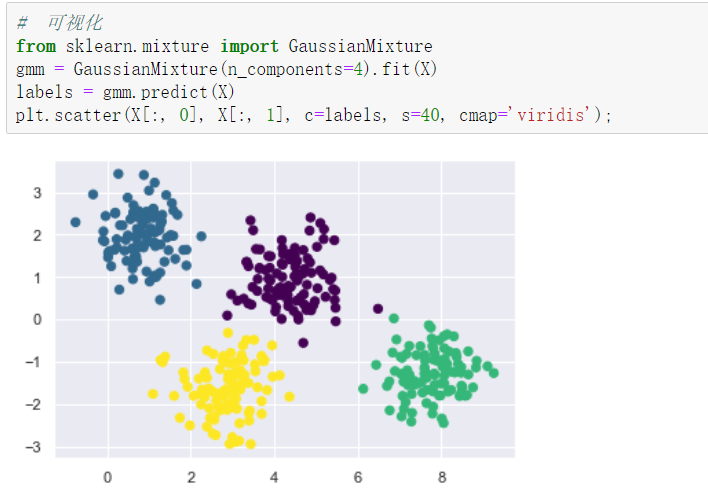
       假设 有K个类，样本数量分别为N1,N2,…,Nk且N1+N2+…+Nk=N，即有观测数据https://images0.cnblogs.com/blog/512363/201304/24162706-afec28010ffa487a8e48975acec86fb3.gif，第k个分类的样本集合表示为S(k)，那么公式（2）中的三个参数可以表示为：

https://images0.cnblogs.com/blog/512363/201304/24162812-657121700b3c4a86a7d515e1f3b403c6.gif

https://images0.cnblogs.com/blog/512363/201304/24162828-e1086e58d5644135b947c4ff274dfc1c.gif

https://images0.cnblogs.com/blog/512363/201304/24162839-e78b7c5f2fa449d6857ec374b168ebeb.gif  
实验结果：



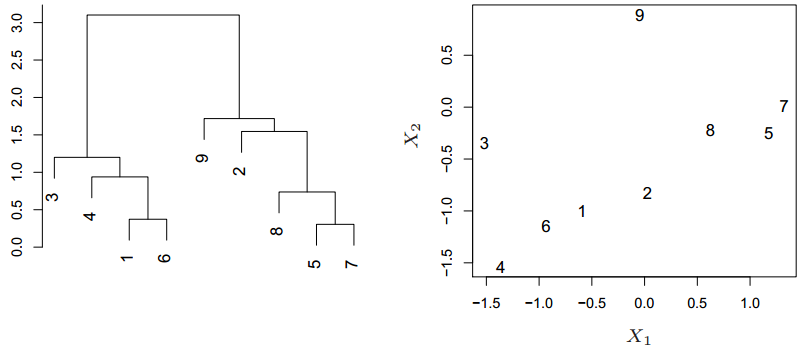


* 层次聚类

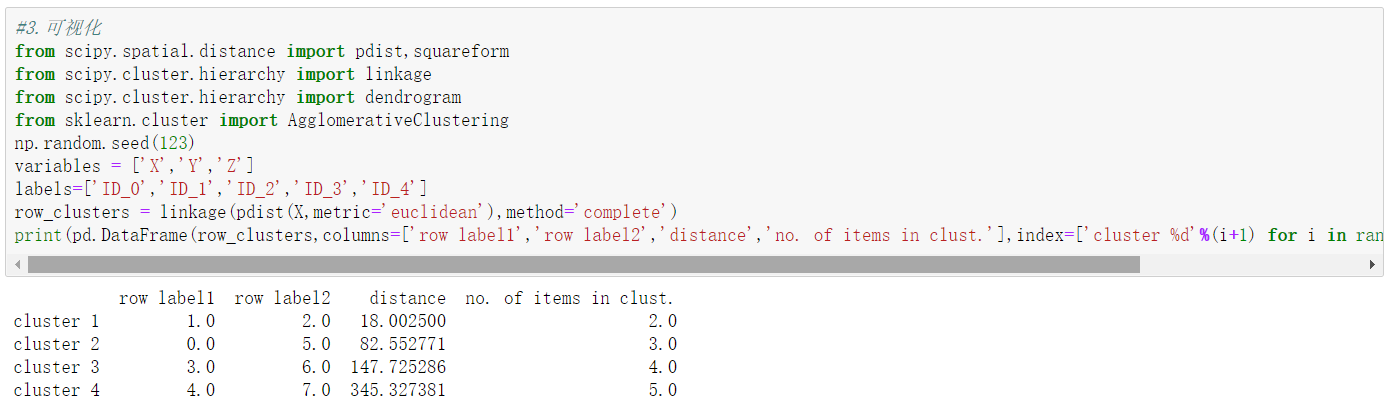
它的基本过程如下：

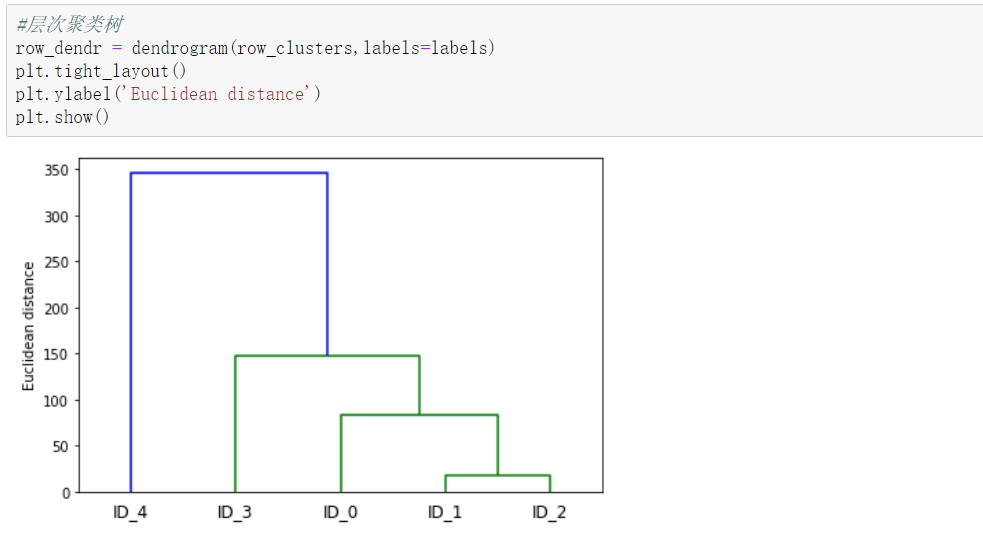
* 每一个样本点视为一个簇；
* 计算各个簇之间的距离，最近的两个簇聚合成一个新簇；
* 重复以上过程直至最后只有一簇。

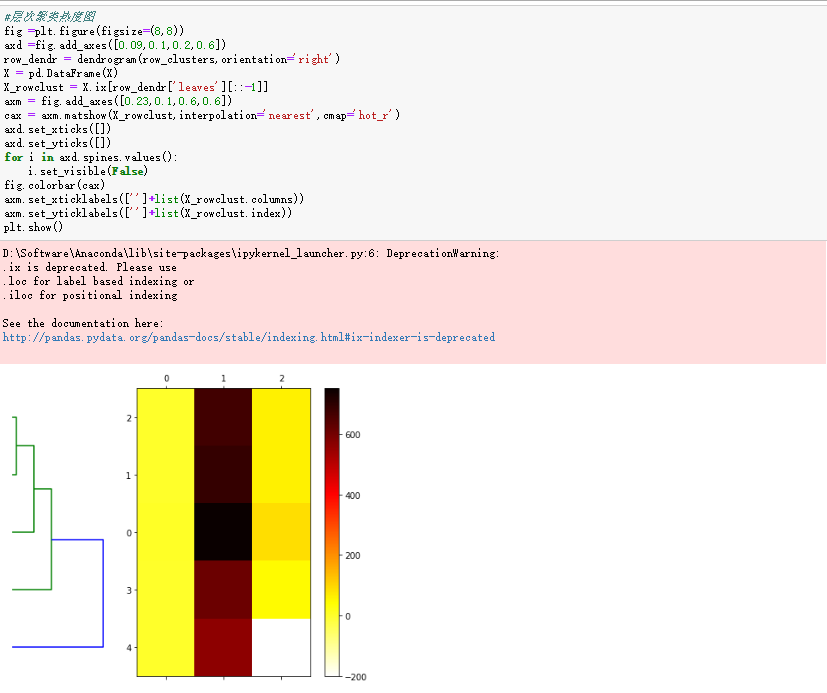
层次聚类不指定具体的簇数，而只关注簇之间的远近，最终会形成一个树形图。



实验结果：







# 数据降维与可视化——t-SNE

  t-SNE是目前来说效果最好的数据降维与可视化方法，但是它的缺点也很明显，比如：占内存大，运行时间长。但是，当我们想要对高维数据进行分类，又不清楚这个数据集有没有很好的可分性（即同类之间间隔小，异类之间间隔大），可以通过t-SNE投影到2维或者3维的空间中观察一下。如果在低维空间中具有可分性，则数据是可分的；如果在高维空间中不具有可分性，可能是数据不可分，也可能仅仅是因为不能投影到低维空间。

### 优化 t-SNE

  t-SNE的主要目的是高维数据的可视化。因此，当数据嵌入二维或三维时，效果最好。有时候优化KL散度可能有点棘手。有五个参数可以控制t-SNE的优化，即会影响最后的可视化质量：

* perplexity困惑度
* early exaggeration factor前期放大系数
* learning rate学习率
* maximum number of iterations最大迭代次数
* angle角度

### Barnes-Hut t-SNE

  Barnes-Hut t-SNE主要是对传统t-SNE在速度上做了优化，是现在最流行的t-SNE方法，同时它与传统t-SNE还有一些不同：

* Barnes-Hut仅在目标维度为3或更小时才起作用。以2D可视化为主。
* Barnes-Hut仅适用于密集的输入数据。稀疏数据矩阵只能用特定的方法嵌入，或者可以通过投影近似，例如使用[sklearn.decomposition.TruncatedSVD](http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html#sklearn.decomposition.TruncatedSVD)
* Barnes-Hut是一个近似值。使用angle参数对近似进行控制，因此当参数method="exact"时，TSNE()使用传统方法，此时angle参数不能使用。
* Barnes-Hut可以处理更多的数据。 Barnes-Hut可用于嵌入数十万个数据点。

  为了可视化的目的（这是t-SNE的主要用处），强烈建议使用Barnes-Hut方法。method="exact"时，传统的t-SNE方法尽管可以达到该算法的理论极限，效果更好，但受制于计算约束，只能对小数据集的可视化。

  对于MNIST来说，t-SNE可视化后可以自然的将字符按标签分开，见本文最后的例程；而PCA降维可视化后的手写字符，不同类别之间会重叠在一起，这也证明了t-SNE的非线性特性的强大之处。值得注意的是：未能在2D中用t-SNE显现良好分离的均匀标记的组不一定意味着数据不能被监督模型正确分类，还可能是因为2维不足以准确地表示数据的内部结构。