**无标签数据的无监督学习-聚类建模算法**

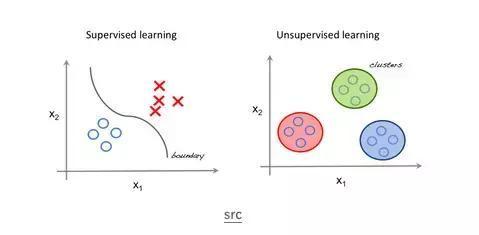
无监督学习是机器学习技术中的一类，用于发现数据中的模式。本文介绍用Python进行无监督学习的几种聚类算法，包括**K-Means聚类、分层聚类、t-SNE聚类、DBSCAN聚类、两步聚类、BIRCH、谱聚类等。**

聚类分析本身不是一个特定的算法，而是要解决的一般任务。它可以通过各种算法来实现，这些算法在理解群集的构成以及如何有效地找到它们方面存在显着差异。流行的群集概念包括群集成员之间距离较小的群体，数据空间的密集区域，间隔或特定的统计分布。因此，聚类可以表述为多目标优化问题。适当的聚类算法和参数设置（包括距离函数等参数）使用，密度阈值或预期聚类的数量）取决于个体数据集和结果的预期用途。这样的聚类分析不是自动任务，而是涉及试验和失败的知识发现或交互式多目标优化的迭代过程。通常需要修改数据预处理和模型参数，直到结果达到所需的属性。

无监督学习是机器学习技术中的一类，用于发现数据中的模式。无监督算法的数据没有标注，这意味着**只提供输入变量（X）**，**没有相应的输出变量**。在无监督学习中，算法自己去发现数据中有意义的结构。

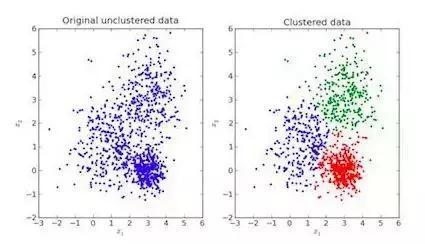
**监督学习 VS 无监督学习**

在监督学习中，系统试图从之前给出的例子中学习。反之，在无监督学习中，系统试图从给出的例子中直接找到模式。因此，如果数据集有标记，那么它是有监督问题，如果数据集无标记，那么它是一个无监督问题。



**左边是监督学习**的例子; 我们使用回归技术来寻找特征之间的最佳拟合线。而在**右边无监督学习**中，输入是基于特征分离的，预测则取决于它属于哪个聚类（cluster）。

使用**Iris数据集（鸢尾花卉数据集）**来进行我们的第一次预测。该数据集包含150条记录的一组数据，有5个属性——花瓣长度，花瓣宽度，萼片长度，萼片宽度和类别。三个类别分别是Iris Setosa（山鸢尾），Iris Virginica（维吉尼亚鸢尾）和Iris Versicolor（变色鸢尾）。对于我们的无监督算法，我们给出鸢尾花的这四个特征，并预测它属于哪一类。我们在Python中使用sklearn Library来加载Iris数据集，并使用matplotlib来进行数据可视化。



在上图中，左边的图像是未完成分类的原始数据，右边的图像是聚类的（根据数据的特征对数据进行分类）。当给出要预测的输入时，就会根据它的特征在它所属的聚类中进行检查，并做出预测。

**一、Python中的K-Means聚类**

K-Means是一种**迭代聚类算法**，它的目的是在每次迭代中找到局部最大值。首先，选择所需数量的聚类。由于我们已经知道涉及3个类，因此我们通过将参数“n\_clusters”传递到K-Means模型中，将数据分组为3个类。

现在，随机将三个点（输入）分成三个聚类。基于每个点之间的质心距离，下一个给定的输入被分为所需的聚类。然后，重新计算所有聚类的质心。聚类的每个质心是特征值的集合，定义生成的组。检查质心特征权重可以定性地解释每个聚类代表什么类型的组。

从sklearn库导入K-Means模型，拟合特征并进行预测。

**k-means要注意数据异常：**

（1）数据异常值。数据中的异常值能明显改变不同点之间的距离相识度，并且这种影响是非常显著的。因此基于距离相似度的判别模式下，异常值的处理必不可少。

（2）数据的异常量纲。不同的维度和变量之间，如果存在数值规模或量纲的差异，那么在做距离之前需要先将变量归一化或标准化。例如跳出率的数值分布区间是[0,1]，订单金额可能是[0,10000 000]，而订单数量则是[0,1000]，如果没有归一化或标准化操作，那么相似度将主要受到订单金额的影响。

**K-means聚类延申：****MiniBatchKMeans**

K均值在算法稳定性、效率和准确率（相对于真实标签的判别）上表现非常好，并且在应对大量数据时依然如此。它的算法时间复杂度上界为O(nkt)，其中n是样本量、k是划分的聚类数、t是迭代次数。当聚类数和迭代次数不变时，K均值的算法消耗时间只跟样本量有关，因此会呈线性增长趋势。

但是当面对海量数据时，k均值算法计算速度慢会产生延时，尤其算法被用于做实时性处理时这种弊端尤为明显。针对K均值的这一问题，很多延伸算法出现了，MiniBatchKMeans就是其中一个典型代表。MiniBatchKMeans使用了一个种名为Mini Batch（分批处理）的方法计算数据点之间的距离。Mini Batch的好处是计算过程中不必使用所有的数据样本，而是从不同类别的样本中抽取一部分样本（而非全部样本）作为代表参与聚类算法过程。由于计算样本量少，所以会相应减少运行时间；但另一方面，由于是抽样方法，抽样样本很难完全代表整体样本的全部特征，因此会带来准确度的小幅度下降，但是并不明显。

大致分为两种：

**K-均值聚类算法：**

1. 加载数据集

2. 数据初始化

2.1 创建随机质心点

2.2 穿件保存结果的各个矩阵/数组

3. 多次迭代 （判断所有点的分类是否发生变化）

3.1 计算所有点的分类

3.2 根据3.1分类结果，重新计算质心点（用属于当前类的数据取平均作为新的质心点）

4. 返回数据

**该算法缺点：**

算法容易收敛到局部最小值，而非全局最小值。（局部最小值指结果还可以，但是并非最好结果，全局最小值时可能的最好结果）

**二分K-均值聚类算法：**

**SSE：** 度量聚类效果的指标（Sum of Squared Erro，**误差平方和**）

SSE越小说明所有数据点越接近他们的质心，聚类效果也就越好。

二分k均值（bisecting k-means）算法的主要思想是：首先将所有点作为一个簇，然后将该簇一分为二。之后选择能最大程度降低聚类代价函数（也就是误差平方和）的簇划分为两个簇。以此进行下去，直到簇的数目等于用户给定的数目k为止。

以上隐含着一个原则是：因为聚类的误差平方和能够衡量聚类性能，该值越小表示数据点月接近于它们的质心，聚类效果就越好。所以我们就**需要对误差平方和最大的簇进行再一次的划分，因为误差平方和越大，表示该簇聚类越不好，越有可能是多个簇被当成一个簇了，所以我们首先需要对这个簇进行划分。**

1. 将所有点看成一个簇

2. 当簇数目小于K时

2.1 对每个簇

2.1.1 计算总误差

2.1.2 在给定簇上面进行K-均值聚类（K=2）

2.1.2 计算在该簇上一分为二之后的总误差

2.2 选择是的误差最小的那个簇进行划分

**二、分层聚类、层次聚类（Hierarchical Clustering）**

### 1）概述

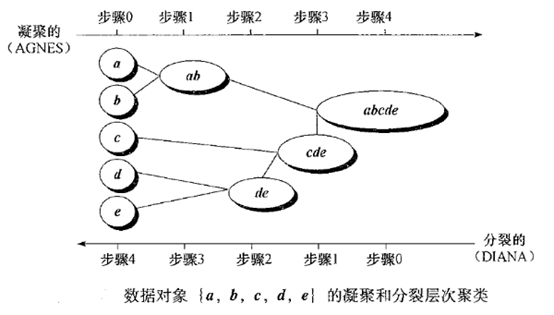
Hierarchical Clustering(层次聚类)：就是按照某种方法进行层次分类，直到满足某种条件为止。

**主要分成两类：**

**a）凝聚：**从下到上。首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到所有的对象都在一个簇中，或者某个终结条件被满足。

**b）分裂：**从上到下。首先将所有对象置于同一个簇中，然后逐渐细分为越来越小的簇，直到每个对象自成一簇，或者达到了某个终止条件。**（较少用）**

分层聚类是一种构建聚类层次结构的算法。该算法从分配给它们自己的一个cluster的所有数据开始，然后将最近的两个cluster加入同一个cluster。最后，当只剩下一个cluster时，算法结束。分层聚类的完成可使用树状图表示。



### 2）算法步骤

a）将每个对象归为一类, 共得到N类, 每类仅包含一个对象. 类与类之间的距离就是它们所包含的对象之间的距离.

b）找到最接近的两个类并合并成一类, 于是总的类数少了一个.

c）重新计算新的类与所有旧类之间的距离.  
d）重复第2步和第3步, 直到最后合并成一个类为止(此类包含了N个对象).

### 3）算法函数

**a）sklearn.cluster.AgglomerativeClustering**

**b）主要参数**([详细参数](http://scikit-learn.org/dev/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering))

* **n\_clusters：聚类的个数**
* **linkage：指定层次聚类判断相似度的方法**，有以下三种：
  + **ward：**组间距离等于两类对象之间的最小距离。（即single-linkage聚类）
  + **average：**组间距离等于两组对象之间的平均距离。（average-linkage聚类）
  + **complete：**组间距离等于两组对象之间的最大距离。（complete-linkage聚类）

c）主要属性

* **labels\_：** 每个数据的分类标签

# ****4）KMeans聚类与分层聚类的区别****

* 分层聚类不能很好地处理大数据，但KMeans聚类可以。因为KMeans的时间复杂度是线性的，即O（n），而分层聚类的时间复杂度是二次的，即O（n2）。
* 在KMeans聚类中，当我们从聚类的任意选择开始时，多次运行算法产生的结果可能会有所不同。不过结果可以在分层聚类中重现。
* 当聚类的形状是超球形时（如2D圆形，3D球形），KMeans聚类更好。
* K-Means聚类不允许嘈杂的数据，而在分层聚类中，可以直接使用嘈杂的数据集进行聚类。

**三、t-SNE聚类**t-SNE聚类是用于可视化的无监督学习方法之一。t-SNE表示t分布的随机

近邻嵌入。它将高维空间映射到可以可视化的2或3维空间。具体而言，它通过

二维点或三维点对每个高维对象进行建模，使得相似的对象由附近的点建模，而

不相似的对象很大概率由远离的点建模。

**四、DBSCAN聚类**

**1）概述**

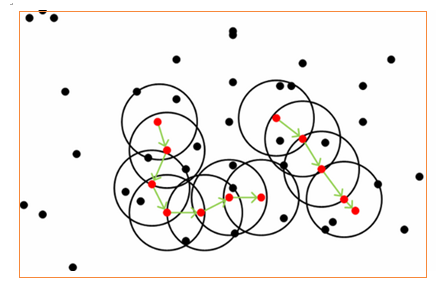
DBSCAN（Density-Based Spatial Clustering of Applications with Noise，具有噪声的基于密度的聚类方法）是一种流行的聚类算法，用作预测分析中 K-means的替代。它不要求输入聚类的数值才能运行。但作为交换，你必须调整其他两个参数。该算法将具有**足够密度的区域划分为簇**(即要求聚类空间中的一定区域内所包含对象的数目不小于某一给定阈值)，并在具有噪声的空间数据库中发现任意形状的簇，它**将簇定义为密度相连的点的最大集合**。

**2）算法步骤：**

DBSCAN需要二个参数:**扫描半径 (eps)和最小包含点数(min\_samples)**

a）遍历所有点，寻找核心点。

b）连通核心点，并且在此过程中扩展某个分类集合中点的个数。

****

第一步就是寻找红色的核心点，第二步就是用绿色箭头联通红色点。图中点以绿色线条为中心被分成了两类。没在黑色圆中的点是**噪声点**。

**3）算法函数：**

**a）sklearn.cluster.DBSCAN**

**b）主要参数（[详细参数](http://scikit-learn.org/dev/modules/generated/sklearn.cluster.DBSCAN.html" \l "sklearn.cluster.DBSCAN)）**

* **eps:**两个样本之间的最大距离，即扫描半径
* **min\_samples ：**作为核心点的话邻域(即以其为圆心，eps为半径的圆，含圆上的点)中的最小样本数(包括点本身)。

**c）主要属性**

* **core\_sample\_indices\_:**核心样本指数。（此参数在代码中有详细的解释）
* **labels\_:**数据集中每个点的集合标签给,噪声点标签为-1。

**4）优缺点比较：**

# **有**异常的数据**可以使用**DBSCAN聚类方法**进行处理，DBSCAN的全称是Density-Based Spatial Clustering of Applications with Noise，中文含义是“基于密度的带有噪声的空间聚类”。跟K均值相比，它具有以下**优点**：**

# **（1）原始数据分布规律没有明显要求，能适应任意数据集分布形状的空间聚类，因此数据集适用性更广，尤其是对非凸装、圆环形等异性簇分布的识别较好。**可以发现任意形状的聚类。****

# **（2）**无需指定聚类数量**，对结果的先验要求不高。**

# **（3）由于DBSCAN可区分核心对象、边界点和噪点，因此对噪声的过滤效果好，能有效应对数据噪点。**

**由于他对整个数据集进行操作且聚类时使用了一个全局性的表征密度的参数，因此也存在比较明显的弱点：**

随着数据量的增加，对**I/O、内存的要求也随之增加**。

如果**密度分布不均匀，聚类效果较差**。

# **（1）对于高纬度问题，基于半径和密度的定义成问题。**

# **（2）当簇的密度变化太大时，聚类结果较差。**

# **（3）当数据量增大时，要求较大的内存支持，I/O消耗也很大。**

# ****五、********谱聚类（Spectral Clustering）****

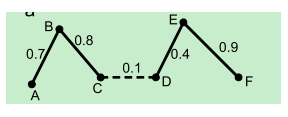
# **在大数据背景下，有很多**高纬度数据场景**，如电子商务交易数据、web文本数据日益丰富。高维数据聚类时耗时长、聚类结果准确性和稳定性都不尽如人意。因为，在高维数据，基于距离的相似度计算效率极低；特征值过多在所有维度上存在簇的可能性非常低；由于稀疏性和紧邻特性，基于距离的相似度几乎为0，导致高维空间很难出现数据簇。这时我们可以选择使用子空间聚类，或是降维处理。**

# ****子空间聚类算法**是在高维数据空间中对传统聚类算法的一种扩展，其思想是选取与给定簇密切相关的维，然后在对应的子空间进行聚类。比如谱聚类就是一种子空间聚类方法，由于选择相关维的方法以及评估子空间的方法需要自定义，因此这种方法对操作者的要求较高。**

**Spectral Clustering(SC,即谱聚类)，是一种基于图论的聚类方法,它能够识别任意形状的样本空间且收敛于全局最优解，其基本思想是利用样本数据的相似矩阵进行特征分解后得到的特征向量进行聚类.它与样本特征无关而只与样本个数有关。**

**基本思路：将样本看作顶点,样本间的相似度看作带权的边,从而将聚类问题转为图分割问题:找到一种图分割的方法使得连接不同组的边的权重尽可能低(这意味着组间相似度要尽可能低),组内的边的权重尽可能高(这意味着组内相似度要尽可能高)。**

**断开虚线，六个数据被聚成两类。**



**Spectral Clustering算法函数**

**a）核心函数：sklearn.cluster.SpectralClustering**

因为是基于图论的算法，所以**输入必须是对称矩阵**。

**b）主要参数**(参数较多，[详细参数](http://scikitlearn.org/dev/modules/generated/sklearn.cluster.SpectralClustering.html" \l "sklearn.cluster.SpectralClustering))

* **n\_clusters：聚类的个数**。（官方的解释：投影子空间的维度）
* **affinity：核函数**，默认是’rbf’，可选：“nearest\_neighbors”

“precomputed”, "rbf"或sklearn.metrics.pairwise\_kernels支持的其中一个内核之一。

* **gamma :affinity指定的核函数的内核系数，默认1.0**

**c）主要属性**

* **labels\_ ：每个数据的分类标签**

# ****六、Mean-shift聚类****

# ****Mean-shift（即：均值迁移）的基本思想：**在数据集中选定一个点，然后以这个点为圆心，r为半径，画一个圆(二维下是圆)，求出这个点到所有点的向量的平均值，而圆心与向量均值的和为新的圆心，然后迭代此过程，直到满足一点的条件结束。后来Yizong Cheng 在此基础上加入了 核函数 和 权重系数 ，使得Mean-shift 算法开始流行起来。目前它在聚类、图像平滑、分割、跟踪等方面有广泛应用。**

# ****Mean-shift 算法核心思想**是不断的寻找新的圆心坐标，直到密度最大的区域。**

# 

# ****Mean-shift 算法函数****

**a）核心函数：sklearn.cluster.MeanShift(核函数：RBF核函数)**

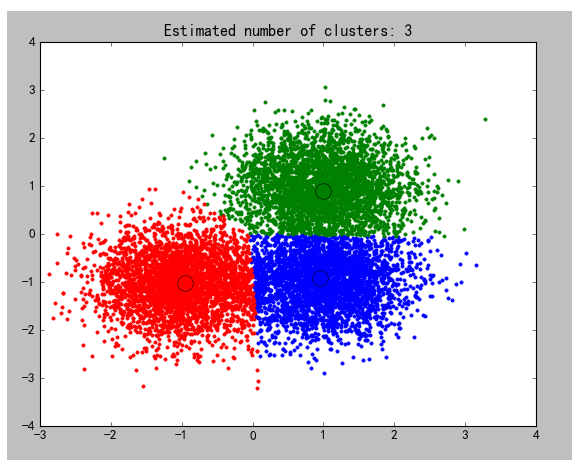
**圆心(或种子)的确定和半径(或带宽)的选择，是影响算法效率的两个主要因素。所以在sklearn.cluster.MeanShift中重点说明了这两个参数的设定问题。**

**b）主要参数**

* **bandwidth ：半径(或带宽)，float型。如果没有给出，则使用sklearn.cluster.estimate\_bandwidth计算出半径(带宽).（可选）**
* **seeds :圆心（或种子），数组类型，即初始化的圆心。（可选）**
* **bin\_seeding ：布尔值。如果为真，初始内核位置不是所有点的位置，而是点的离散版本的位置，其中点被分类到其粗糙度对应于带宽的网格上。将此选项设置为True将加速算法，因为较少的种子将被初始化。默认值：False.如果种子参数(seeds)不为None则忽略。**

**c）主要属性**

* **cluster\_centers\_ : 数组类型。计算出的聚类中心的坐标。**
* **labels\_ :数组类型。每个数据点的分类标签。**



**OpenCV主要应用于图像处理，而Mean-shift多用于图像跟踪等，所以对应图像处理这部分而言，openCV中的Mean-shift算法的功能还是强大一点。**

# ****七． Birch聚类方法****

### 1）概述

Birch(利用**层次方法的平衡迭代规约和聚类**)：就是**通过聚类特征(CF)形成一个聚类特征树**，**root层的CF个数就是聚类个数。**

### 2）相关概念：

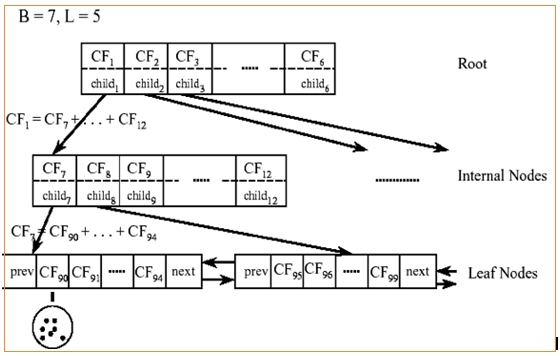
**聚类特征(CF)：**每一个CF是一个三元组,可以用（N，LS，SS）表示.其中**N**代表了这个CF中拥有的**样本点数量**;**LS**代表了这个CF中拥有的**样本点各特征维度的和向量**,**SS**代表了这个CF中拥有的**样本点各特征维度的平方和**。

# 

如上图所示：N = 5

LS=(3+2+4+4+3, 4+6+5+7+8)=(16, 30)

SS =(32+22+42+42+32,42+62+52+72+82)=(54,190)



# 对于上图中的CF Tree,限定了B=7,L=5， 也就是说内部节点最多有7个CF(CF90下的圆), 而叶子节点最多有5个CF(CF90到CF94)。叶子节点是通过双向链表连通的。

# ****3）算法函数****

# a）sklearn.cluster.Birch

**b）主要参数（[详细参数](http://scikit-learn.org/dev/modules/generated/sklearn.cluster.Birch.html" \l "sklearn.cluster.Birch)）**

* **n\_clusters ：**聚类的目标个数。（可选）
* **threshold ：**扫描半径（个人理解，官方说法比较绕口），设置小了分类就多。
* **branches\_factor：**每个节点中CF子集群的最大数量,默认为50。

**c）主要属性**

* labels\_ ：每个数据点的分类

# ****八．** GaussianMixtureModel**聚类方法****

### 1）概述

正太分布也叫高斯分布，正太分布的概率密度曲线也叫高斯分布概率曲线。

GaussianMixtureModel(混合高斯模型，GMM)。

聚类算法大多数通过相似度来判断，而相似度又大多采用欧式距离长短作为衡量依据。而GMM采用了新的判断依据：概率，即通过属于某一类的概率大小来判断最终的归属类别。

GMM的基本思想就是：任意形状的概率分布都可以用多个高斯分布函数去近似，也就是说GMM就是有多个单高斯密度分布（Gaussian）组成的，每个Gaussian叫一个"Component"，这些"Component"线性加成在一起就组成了 GMM 的概率密度函数，也就是下面的函数。



**K：模型的个数**，即Component的个数（**聚类的个数**）

**πk：为第k个高斯的权重**

**p（x |k）：**为第k个高斯概率密度,其均值为μk,方差为σk

上述参数，除了K是直接给定之外，其他参数都是通过EM算法估算出来的。(有个参数是指定EM算法参数的)

**（3）GaussianMixtureModel 算法函数**

**a）from sklearn.mixture.GaussianMixture**

**b）主要参数（[详细参数](http://scikit-learn.org/dev/modules/generated/sklearn.mixture.GaussianMixture.html" \l "sklearn.mixture.GaussianMixture)）**

* n\_components ：高斯模型的个数，即聚类的目标个数
* covariance\_type : 通过EM算法估算参数时使用的协方差类型，默认是"full"
* full：每个模型使用自己的一般协方差矩阵
* tied：所用模型共享一个一般协方差矩阵
* diag：每个模型使用自己的对角线协方差矩阵
* spherical：每个模型使用自己的单一方差

**更多无监督技术：**

* 主成分分析（PCA）
* 异常检测（Anomaly detection）
* 自动编码（Autoencoders）
* 深度置信网络（Deep Belief Nets）
* Hebbian Learning
* 生成对抗网络（GAN）
* 自组织映射（Self-Organizing maps）

**使用聚类分析进行中间过程的预处理：**

**图像压缩**

**用较少的数据量来表示原有的像素矩阵的过程，这个过程称为图像编码。数据图像的显著特点是数据量庞大，需要占用相当大的储存空间，这给图像的存储、计算、传输等带来了不便。因此，现在大多数数字网络下的图像都会经过压缩后再做进一步应用，图像压缩的方法之一便是聚类算法。**

**在使用聚类算法做图像压缩时，我们会定义K个颜色数（例如128种颜色），颜色数就是聚类类别的数量；K均值聚类算法会把类似的颜色分别放在K个簇中，然后每个簇使用一种颜色来代替原始颜色，那么结果就是有多少个簇，就生成了多少种颜色构成的图像，由此实现图像压缩。**

**图像分割**

**图像分割就是把图像分成若干个特定的、具有独特性质的区域并提出感兴趣的目标技术和过程，这是图像处理和分析的关键步骤。图像分割后提取出的目标可以用于图像语义识别，图像搜索等领域。例如从图像中分割出前景人脸信息，然后做人脸识别。聚类算法是图像分割方法的一种，其实施的关键是通过不同区域间明显不同的图像色彩特征做聚类，聚类数量就是要分割的区域的数量。**

**图像理解**

**在图像理解中，有一种称为基于区域的提取方法。基于区域的提取方法是在图像分割和对象识别的前提下进行的，利用对象模板、场景分类器等，通过识别对象及对象之间的拓扑关系挖掘语义，生成对应的场景语义信息。例如，先以颜色、形状等特征对分割后的图像区域进行聚类，形成少量BLOB；然后通过CMRM模型计算出BLOB与某些关键词共同出现的概率。**

**异常检测**

**异常检测有多种实施方法，其中常用的方法是基于距离的异常检测方法。即使数据集不满足任何特定分布模型，它仍能有效地发现离群点，特别是当空间维度比较高时，算法的效率比基于密度的方法要高得多。算法具体实现时，首先算出数据样本间的距离（如曼哈顿距离、欧氏距离等），然后对数据做预处理后就可以根据距离的定义来检测异常值。例如，可以使用K-means的聚类可以将离中心店最远的类或者不属于任何一个类的数据点提取出来，然后将其定义为异常值。**

**聚类算法的选择：**

**1）数据为高维数据，那么选取子空间聚类（如谱聚类）**

**2）数据量在100万条以内，那么使用k均值较好；如果数据量超过100万条，那么可以考虑使用Mini Batch KMeans**

**3）如果数据中存在噪点，那么可以使用基于密度的DBSCAN**

**4）如果最求更高的分类准确度，那么选择谱聚类将比K均值准确度更好**