目录

第 6 章

无监督学习：使用未标记的数据

无监督学习是机器学习的一种，它通过推断出描述未标记，未分类数据的隐藏结构的函数来获得一些观点（insight）。训练的观察量中不包括分类。因此，对学习者没有正确或错误的评估，也没有对所用算法输出的观点的准确性进行评估。

一方面，无监督学习算法可以执行比有监督学习算法更复杂的处理任务。另一方面，与其他机器学习模型相比，无监督学习可能更不可预测。

算法

我将讨论一些在数据处理中使用的无监督学习的核心算法和常用算法。本书将从最常见的算法之一开始介绍。

K-最近邻算法

球树（BallTree）用于快速泛华的 N 点问题。

打开 Jupyter 软件中的相同示例：Chapter-006A-001-k-Nearest-Neighbor.ipynb

这个例子考虑的是最近的四个邻居和八个邻居。

使用 ball_tree 引擎来处理最近邻居的问题需要有以下过程。

```
nbrs1 = NearestNeighbors(n_neighbors=ncnt1, algorithm='ball_tree' ).fit(X)
```

您的结果如图 6-1（四个邻居）和图 6-2（八个邻居）所示。

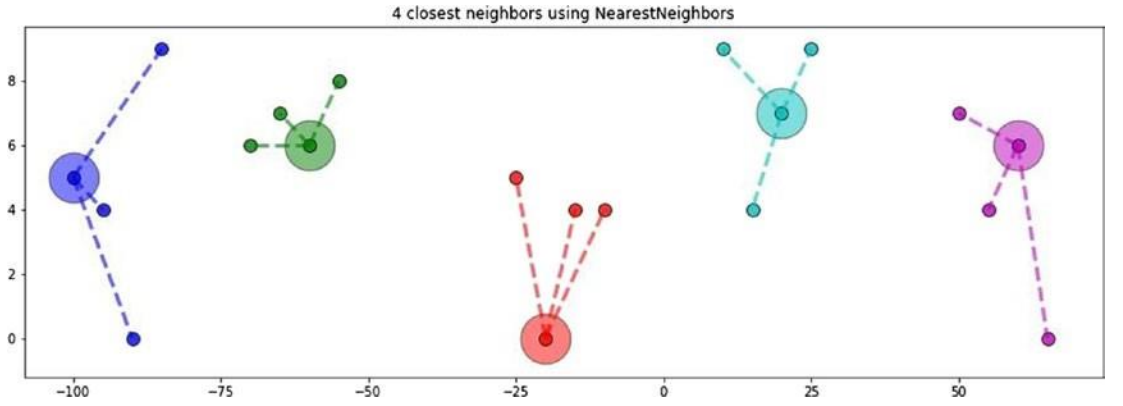


图6-1。四个最近的邻居

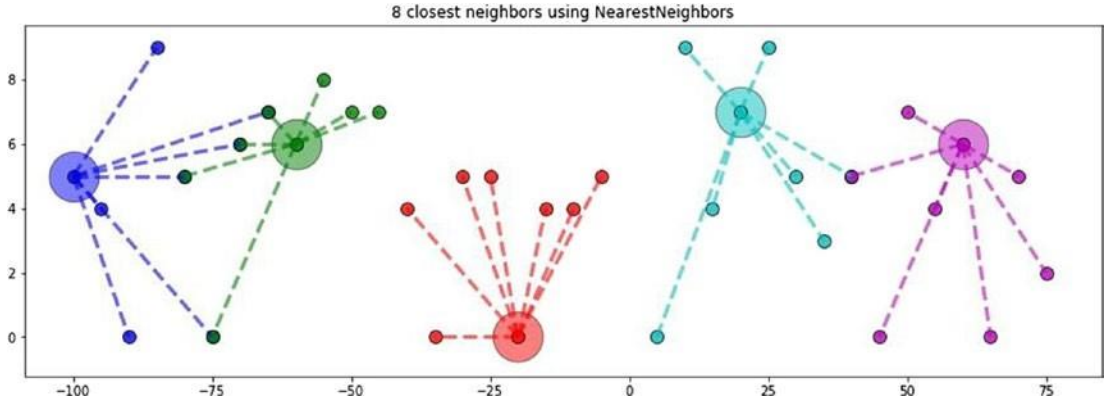


图6-2。八个最近的邻居

您可以观察到在四个邻居的样例中，各点之间没有共同的邻居，但当扩展到八个邻居时，开始观察到有些点有共同的邻居。这种关于共同邻居的调查可以用于电子商务，以决定你会向谁推荐产品或者与在你的区域内，你会和谁联系，也可用在医疗健康上，来判断在令人讨厌的冬季，你可能会传染给谁流感。

请试一试 Jupyter Notebook 的实例：Chapter-006B- 001-k-Nearest-Neighbor.ipynb 和 Chapter-006C- 001-k-Nearest-Neighbor.ipynb，以观察如何将同一算法应用于英国和美国的邮政编码（postcode/zip）以解决问题。（美国邮政编码为 zip，英国为 postcode）

还有一些示例：Chapter-006D- 001-k-Nearest-Neighbor.ipynb 和 Chapter-006E- 001-k-Nearest-Neighbor.ipynb。你能确定这些地点在世界上的位置吗？

确定离关注中心最近的点（最近邻居）的能力是 IML 中用于机电一体化领域的一种能力，来确定最少的资源或者最近的工作区域。

在第 15 章中，我将演示如何使用它来确定采矿地点。

聚类 K - means

沃罗诺伊元胞

K-means 聚类是一种矢量量化方法，最初来自信号处理，在数据挖掘的聚类分析中非常流行。K-means 聚类旨在将 n 个观测值划分为 k 个类，其中每个观测值属于具有最近均值的类，作为聚类的原型。这将导致将数据空间划分为一个个 Voronoi 单元。

打开 Jupyter 实例：Chapter-006-002-Clustering-K-Means-01-02.ipynb

我将向您展示如何将数据划分为 5 个类（参见图 6-3）。

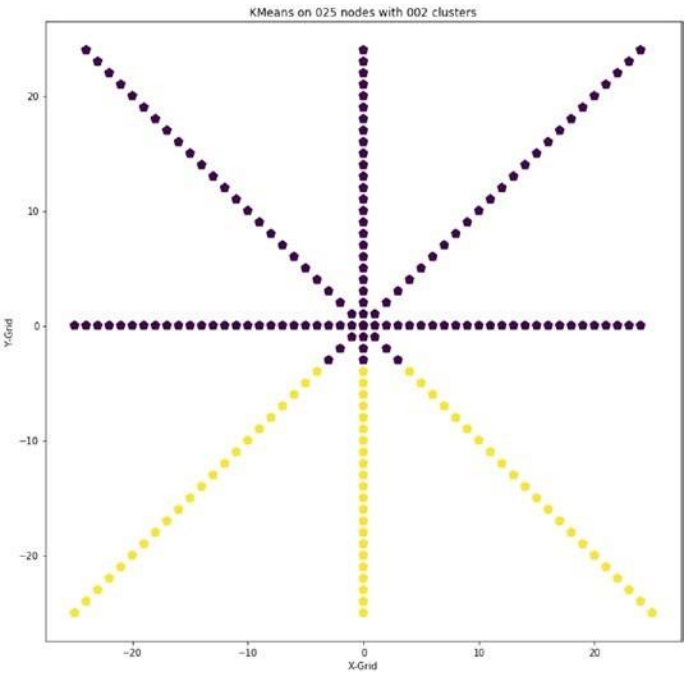


图6-3。 2 个类（簇）的交叉网格

K-Means 可以把数据分成两个类别。

警告：此数据集上的 K-Means 不会 100% 拆分为两个群集。您得到以下结果：

(k-means++) Cluster 1: 1450 (58.0000%) vs. Cluster 2: 1050 (42.0000%)
(random) Cluster 1: 1050 (42.0000%) vs. Cluster 2: 1450 (58.0000%)

这表明，由于机器学习算法的起始状态不同，这两种算法会导致不同的结果。

让我们来研究这两个集群构成的沃罗诺伊细胞（参见图 6-4）。

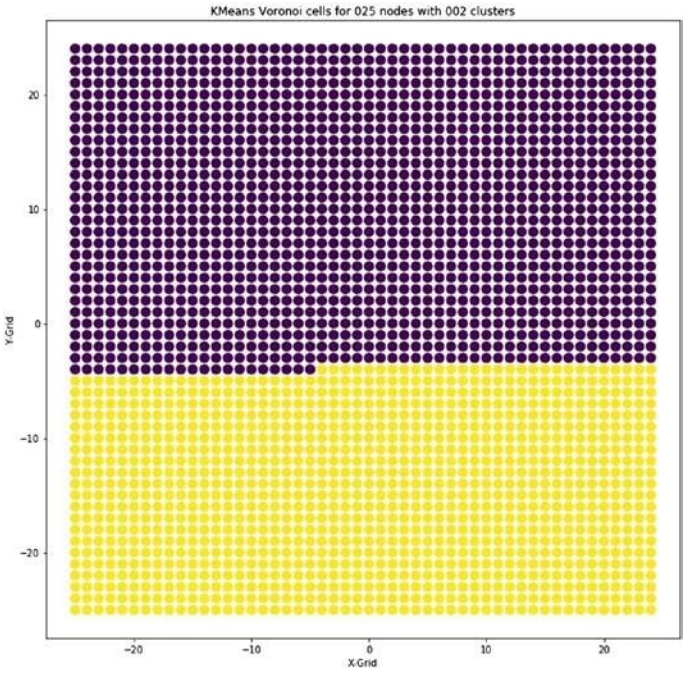


图6-4。 两个沃罗诺伊细胞

这些结果清楚地显示，两个类形成 2 个沃罗诺伊细胞。

如果要试验不同的簇的数量，可以修改此代码中的簇的定义：

N=int（2） 表示需要的簇的数量。

M=int（25）表示网格中的节点。

如果您打开示例：Chapter-006-002-Clustering-K-Means-01-13. ipynb

本次研究的基础是：

N=int （13）

M=int （25）

因此，我们预计有 13 个簇。您的结果如图 6-5 和图 6-6。

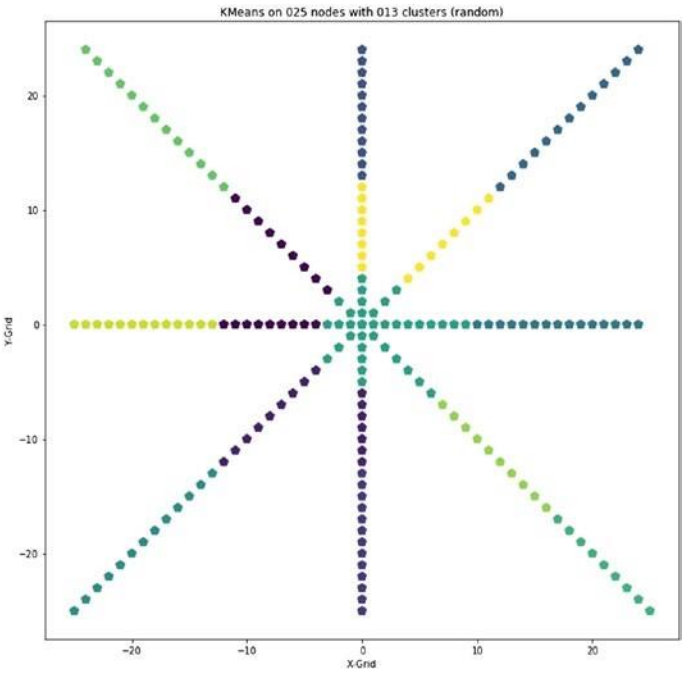


图6-5。 13 个簇的交叉网格

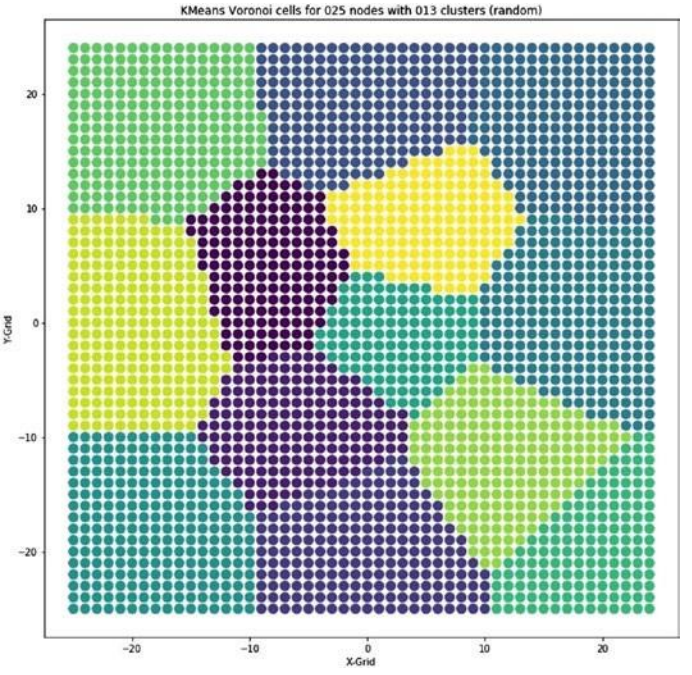


图6-6。 13 个沃罗诺伊细胞

我建议你观察以下见解：

在训练期间，簇如以下列表所示：

Cluster 1 : 42 (21.000 %)
Cluster 2 : 10 (5.000 %)
Cluster 3 : 12 (6.000 %)
Cluster 4 : 16 (8.000 %)
Cluster 5 : 19 (9.500 %)
Cluster 6 : 13 (6.500 %)
Cluster 7 : 9 (4.500 %)
Cluster 8 : 13 (6.500 %)
Cluster 9 : 16 (8.000 %)
Cluster 10 : 12 (6.000 %)
Cluster 11 : 10 (5.000 %)
Cluster 12 : 10 (5.000 %)
Cluster 13 : 18 (9.000 %)

Cluster : 200

但是，当沃罗诺伊细胞被测试时，你会发现一个不同的分布：

Cluster 1 : 142 (5.680 %)

Cluster 2 : 173 (6.920 %)

Cluster 3 : 209 (8.360 %)

Cluster 4 : 227 (9.080 %)

Cluster 5 : 166 (6.640 %)

Cluster 6 : 226 (9.040 %)

Cluster 7 : 149 (5.960 %)

Cluster 8 : 240 (9.600 %)

Cluster 9 : 234 (9.360 %)

Cluster 10 : 204 (8.160 %)

Cluster 11 : 187 (7.480 %)

Cluster 12 : 186 (7.440 %)

Cluster 13 : 157 (6.280 %)

Cluster : 2500

沃罗诺伊细胞是模型的真实预测的分布。

我已经生成了少量其他组合，您可以进行测试：

- Chapter-006-002-Clustering-K-Means-01-04.ipynb
- Chapter-006-002-Clustering-K-Means-01-07.ipynb
- Chapter-006-002-Clustering-K-Means-01-09.ipynb

这些都显示了分布如何随群集的增加而变化。下一步是更改以观察更大的群集过程。

Chapter-006-002-Clustering-K-Means-01-20.ipynb

N=int(20)

M=int(100)

您将观察这两种算法如何给出经过更改的 results：

Cluster 1 : (k-means++) 1653 (4.133 %) vs (random) 1356 (3.390 %) (Net: 297)
Cluster 2 : (k-means++) 1342 (3.355 %) vs (random) 2663 (6.657 %) (Net: -1321)
Cluster 3 : (k-means++) 2929 (7.322 %) vs (random) 2649 (6.623 %) (Net: 280)
Cluster 4 : (k-means++) 2667 (6.667 %) vs (random) 2679 (6.698 %) (Net: -12)
Cluster 5 : (k-means++) 1238 (3.095 %) vs (random) 2996 (7.490 %) (Net: -1758)
Cluster 6 : (k-means++) 1770 (4.425 %) vs (random) 1478 (3.695 %) (Net: 292)
Cluster 7 : (k-means++) 2657 (6.643 %) vs (random) 2556 (6.390 %) (Net: 101)
Cluster 8 : (k-means++) 2704 (6.760 %) vs (random) 1378 (3.445 %) (Net: 1326)
Cluster 9 : (k-means++) 2759 (6.897 %) vs (random) 1329 (3.322 %) (Net: 1430)
Cluster 10 : (k-means++) 2602 (6.505 %) vs (random) 2364 (5.910 %) (Net: 238)
Cluster 11 : (k-means++) 1449 (3.623 %) vs (random) 1077 (2.692 %) (Net: 372)
Cluster 12 : (k-means++) 2668 (6.670 %) vs (random) 1711 (4.277 %) (Net: 957)
Cluster 13 : (k-means++) 2153 (5.382 %) vs (random) 1123 (2.808 %) (Net: 1030)
Cluster 14 : (k-means++) 2668 (6.670 %) vs (random) 2400 (6.000 %) (Net: 268)

Cluster 15 : (k-means++) 1569 (3.923 %) vs (random) 1514 (3.785 %) (Net: 55)
Cluster 16 : (k-means++) 1528 (3.820 %) vs (random) 2701 (6.753 %) (Net: -1173)
Cluster 17 : (k-means++) 1304 (3.260 %) vs (random) 2360 (5.900 %) (Net: -1056)
Cluster 18 : (k-means++) 1711 (4.277 %) vs (random) 1588 (3.970 %) (Net: 123)
Cluster 19 : (k-means++) 1286 (3.215 %) vs (random) 2672 (6.680 %) (Net: -1386)
Cluster 20 : (k-means++) 1343 (3.357 %) vs (random) 1406 (3.515 %) (Net: -63)

Cluster : 40000

提示：经常测试沃罗诺伊细胞，以确保您得到模型的真实结果

现在，我们将根据真正的解决方案来测试我们的新知识。
接下来，我将引导您完成一个示例，该示例包含来自实际 聚类问题的数据。
数据集具有以下结构：
玫瑰数据集是一个特征和无压力的多类分类数据集。

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

打开示例：Chapter-06-003-Clustering-K-Means-02.ipynb
如果执行这个代码，您将注意到该过程创建一系列 K-Means 引擎并将它们作为估计器（参见 C 部分）。
您的结果由 D 部分生成（图 6-7 至 6-10）。

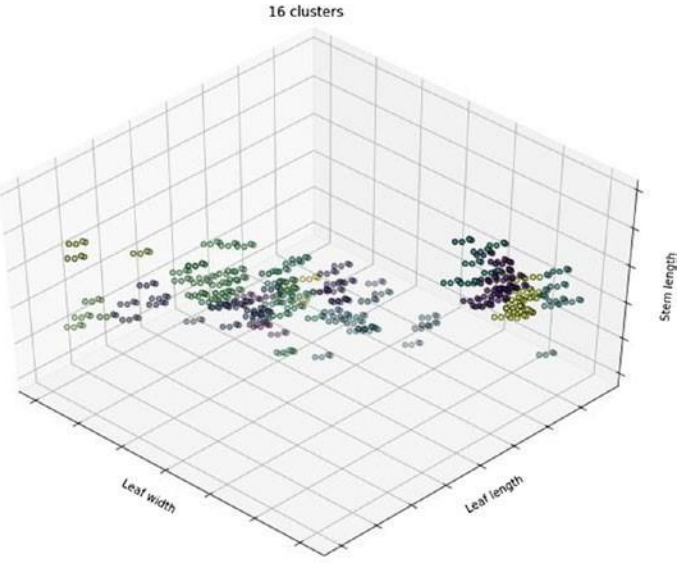


图6-7。 分成16个簇的玫瑰

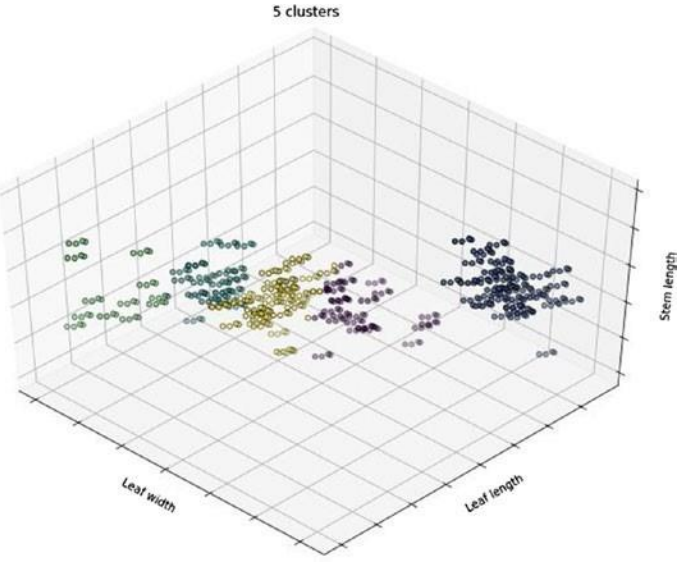


图6-8。 分成5簇的玫瑰

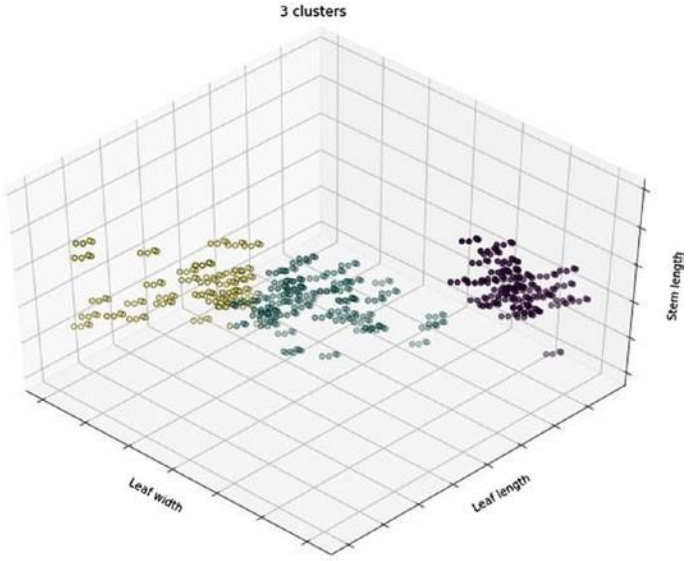


图6-9。 分成3簇的玫瑰

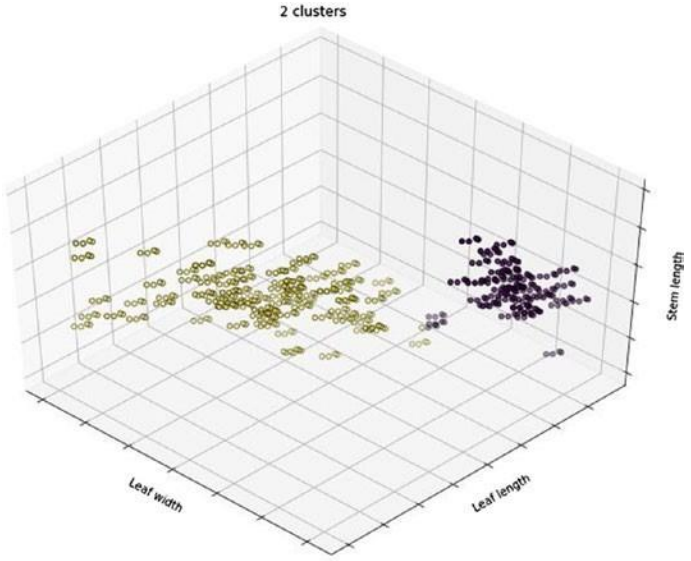


图6-10。 两簇玫瑰

实际数据图显示， 分成三个簇的情况最接近数据的实际分布。

现在，您可以关闭这个程序，因为我们现在转向 K-Means 引擎的另一个常见应用。

现在，我将向您展示如何使用 K-Means 这一新知识来巧妙地处理名为"China.jpg"的示例图像。

打开示例：Chapter-006-004A-Clustering-K-Means-China.ipynb

运行代码以观察 K-Means 引擎如何用于减少图片上的颜色。

有关更多结论，您应该研究 Chapter-006-004B-Clustering-K-MeansFlower.ipynb

你能看出如何使用 K - means 来辅助处理图像吗？

现在你对这一过程已经有了基本的了解，我建议我们尝试使用这种技术对一个著名的地标，即爱丁堡城堡，进行处理。

打开示例： Chapter-006-004C-Clustering-K-MeansEdinburgh.ipynb

爱丁堡的照片可以清楚地展示 K-Means 引擎如何与图像进行交互。

您应该获得以下结果，如图 6-11 到 6-13 所示。

图 6-11。 原始图片

图6-12。 有 64 种颜色的图片

图6-13。 量化后的图片

这种类型的分析在处理 CCTV 图像或机器人视觉时非常有用。

如果您想继续练习，可以研究示例 Chapter-006-004D-Clustering-KMeans-Horses.ipynb

您将注意到，您将首先执行几个次要的图像处理任务，以找到要处理的马。这在实际应用的程序中十分常见。

最佳 K 值

您需要的下一个重点是：您如何决定最优 K 值？

打开文件：Chapter-006-005A-Clustering-K-Means-Optimum. ipynb

找到最优 k 的解决方案是执行 肘部（Elbow ）方法，该方法为给定数据集提供了一个合适的集群（簇 cluster）数量

我已经给你提供了两个数据集：阿尔法（alpha）和贝塔（beta）。我建议您绘制和方差（SSE）和数据集 Alpha 的 K 之间的曲线图。

请注意，本示例的成本函数为"和方差"。SSE 是每个点与其被分配的簇的均值之间的误差的平方和。这是测量给定数据簇内变化的常见方法。本示例使用可接受的成本与给定 的 K 值进行比较，以寻找最优 K 值（图 6-14）。

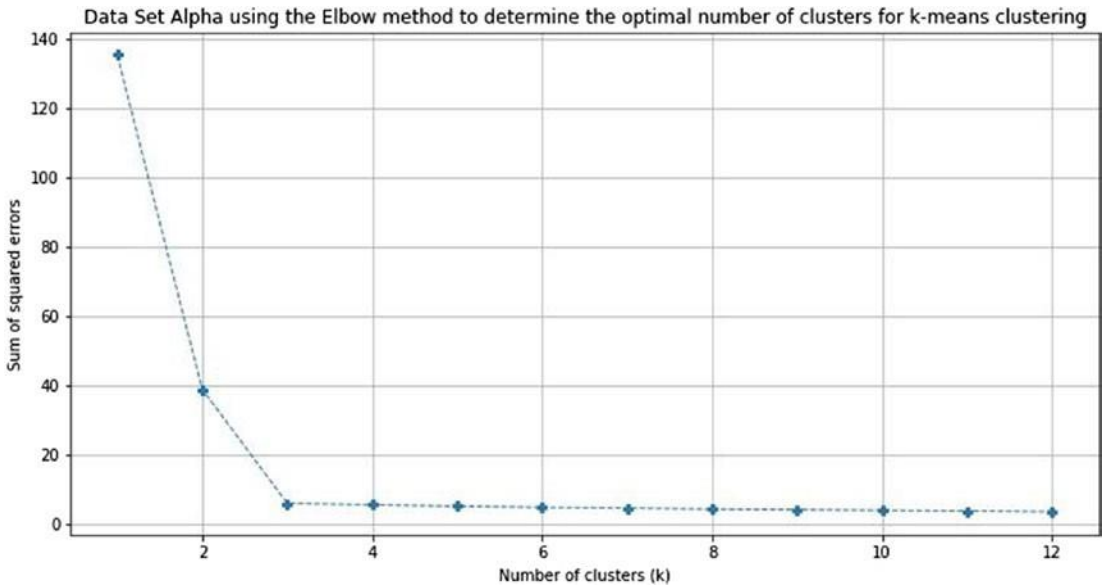


图6-14。 cost 和 K 的函数图，以找到最优 K

Alpha 数据集的最优 K 为 3，即对应图中曲线的肘部。

我建议你看一看数据集 Bata 的 K 值与和方差的曲线（图 6-15）。

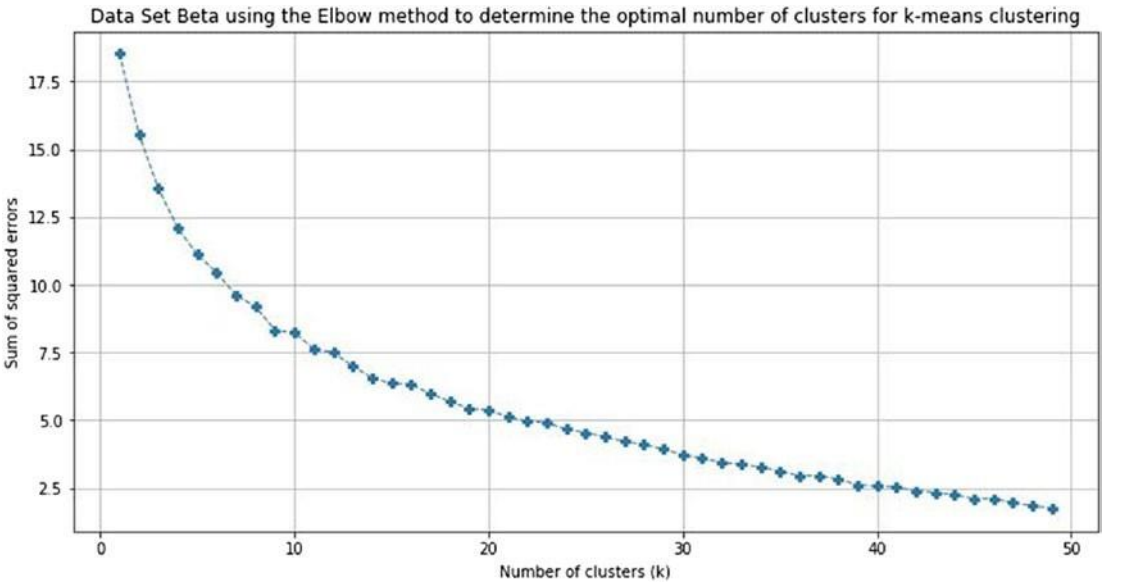


图6-15。 数据集Bata 的 K 与 SSE

曲线的肘部不购明显意味着最优聚类并不明显。

现在，您可以使用这种肘部方法确定数据集的最优 k 是多少。

恭喜你：现在，您可以解决几个 K-Means 问题。

在第 11、12、13、14 和 15 章中，我将指导您学习一些其他的例子，说明如何使用 K-Means 的工业化版本来解决现实世界的问题。

现在，我将引导您学习第二个无监督学习的算法。

高斯混合模型

高斯混合模型是一个假设所有的数据点都是生成于一个混合的有限数量的并且未知参数的高斯分布的概率模型。 我们可以将混合模型看作是 k-means 聚类算法的推广，它利用了关于数据的协方差结构以及潜在高斯中心的信息。

例如：

打开代码： Chapter-006-006-Gaussian-Mixture-01.ipynb

您将使用玫瑰系列 7 数据集来理解高斯模型。

这个解使用了四个不同的协方差参数。

- “full”-每个分量有各自不同的标准协方差矩阵
- “tied”-所有分量有相同的标准协方差矩阵。
- “diag”-每个分量有各自不同对角协方差矩阵。
- “spherical”-指每个分量有各自不同的简单协方差矩阵

您的结果如图 6-16 所示：

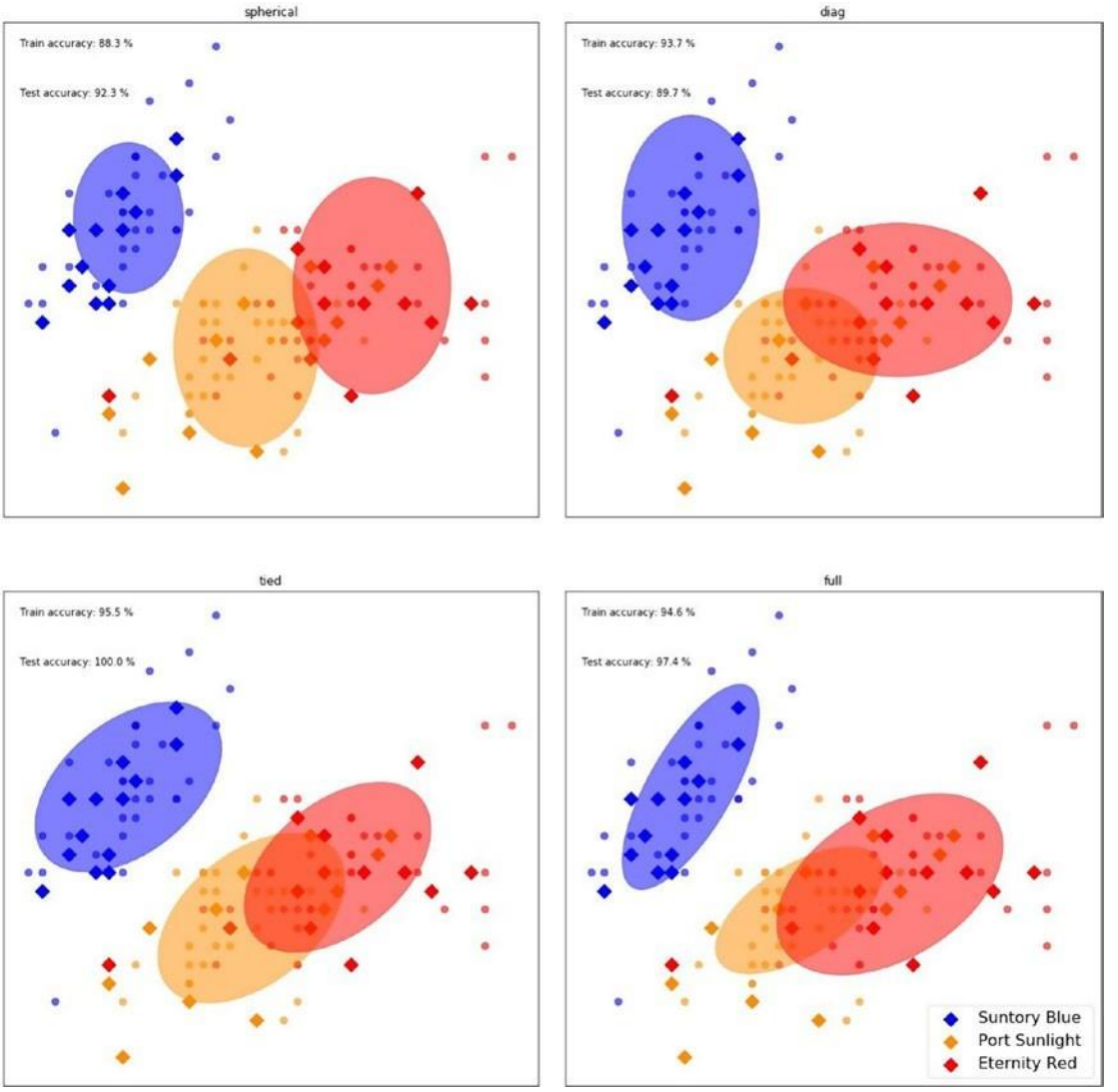


图 6-16。 高斯混合模型的结果

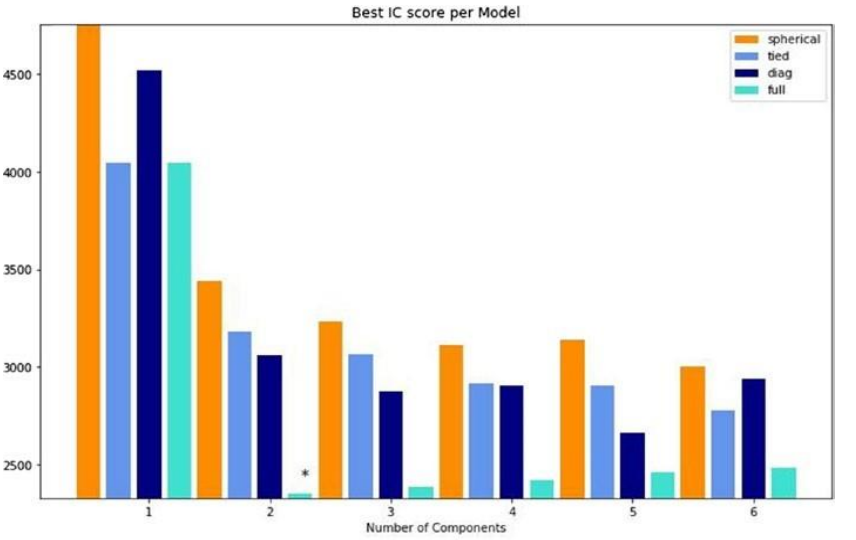
您能了解到不同的协方差类型如何影响聚类的结果吗？

提示　我建议应该每次都在数据集上应用这几种协方差类型，因为我发现有几次即使最轻微的变化也会提供更好的结果。

打开示例代码：Chapter-006-007-Gaussian-Mixture-02.ipynb

这个代码的目的是查找给定数据集的最优协方差参数。

结果如图 6-17。



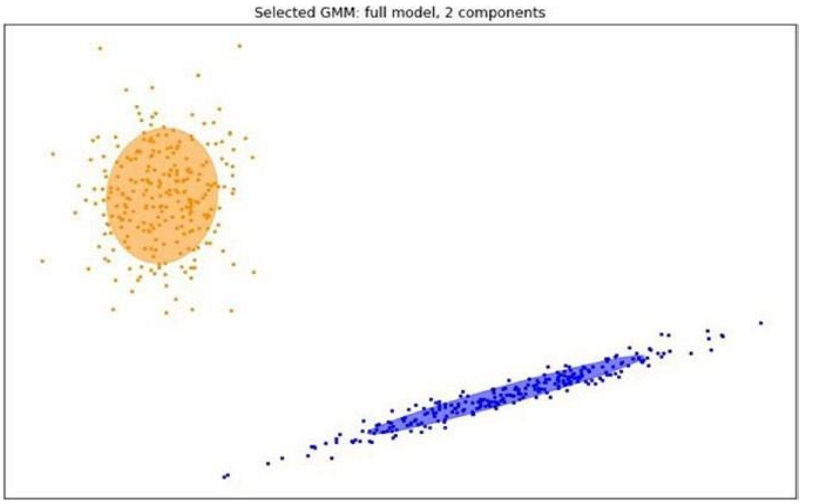


图6-17. 最优高斯混合模型结果

非常好！您现在可以解决高斯混合模型问题。

在第 11、12、13、14 和 15 章中，我将介绍几个额外的例子，说明如何使用高斯混合模型的工业化版本来解决现实世界的问题。

现在，我将介绍第三个无监督学习的算法。

层次聚类

层次聚类（也称为层次聚类分析）是一种将相似对象分成一组的算法。最终停止时产生一个簇的集合，其中每个簇都是不同的，并且每个簇中的对象大致相似。

打开代码：Chapter-006-008-Hierarchical-clustering-01. ipynb

Affinity 是用于计算连接程度（linkage 其实就是距离）的指标，可以从 “euclidean”, “l1”, “l2”, “manhattan”, “cosine”, or ‘precomputed’中选择。

选择要使用的 Linkage。这一标准控制观测集之间要使用的距离类型。该算法将最小化此标准来对合并簇。

- Ward 合并后总方差最小的两个簇。
- average 使用两个集合中观测值的距离的平均数。
- Complete 和 maximum 使用两组观测值之间的最大距离。
- single 使用两组所有观测值之间的最小距离。

结果如图 6-18 所示。

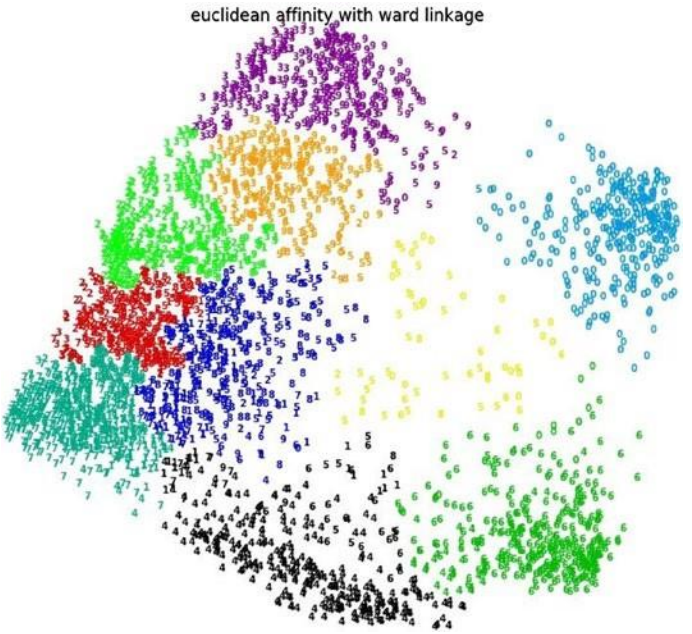


图6-18。 层次聚类（1/8）

聚类的这些参数表示这些参数中的差异对聚类的成功具有好坏两方面的影响。

我建议我们用更可控的方式研究这种影响。

让我们来研究一下 linkage（连接程度）和 affinity（喜好）的影响。

打开代码：Chapter-009-009-Hierarchical-clustering-02. ipynb, Chapter-009-010-Hierarchical-clustering-02.ipynb 和 Chapter-009-011- Hierarchical-clustering-02.ipynb

您将使用嘈杂的圆圈、嘈杂的月亮、斑点，和没有结构的数据配置来生成一个数据集，您可以在其中测试不同超参数的影响。

提示 我个人使用此测试来选择 IML 模型，因为它能很快地显示特定模式是如何影响您的结果的。

执行以上代码并观察结果。

不同的参数会造成一些有趣的，违反直觉的结果。

这是人类在通常情况下仍然更擅长识别复杂模式的主要原因。

我发现最好的组合是人机回圈（human-in-the-loop）的方法。让机器学习首先执行分析，然后调查找到的模式，并根据需要进行人工干预。

您的结果如图 6-19 到图 6-21。

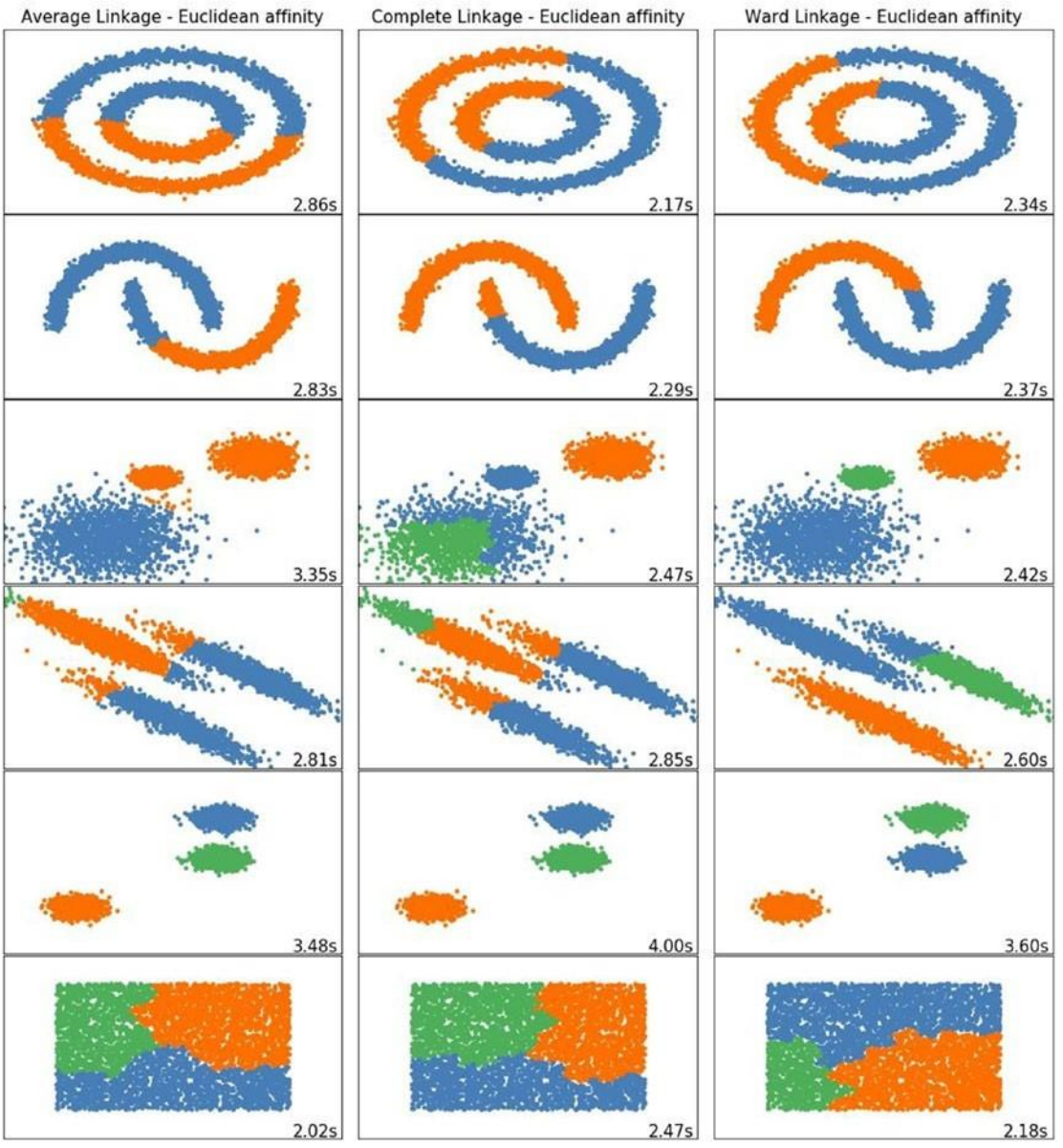


图6-19。 层次 – 连接方式影响

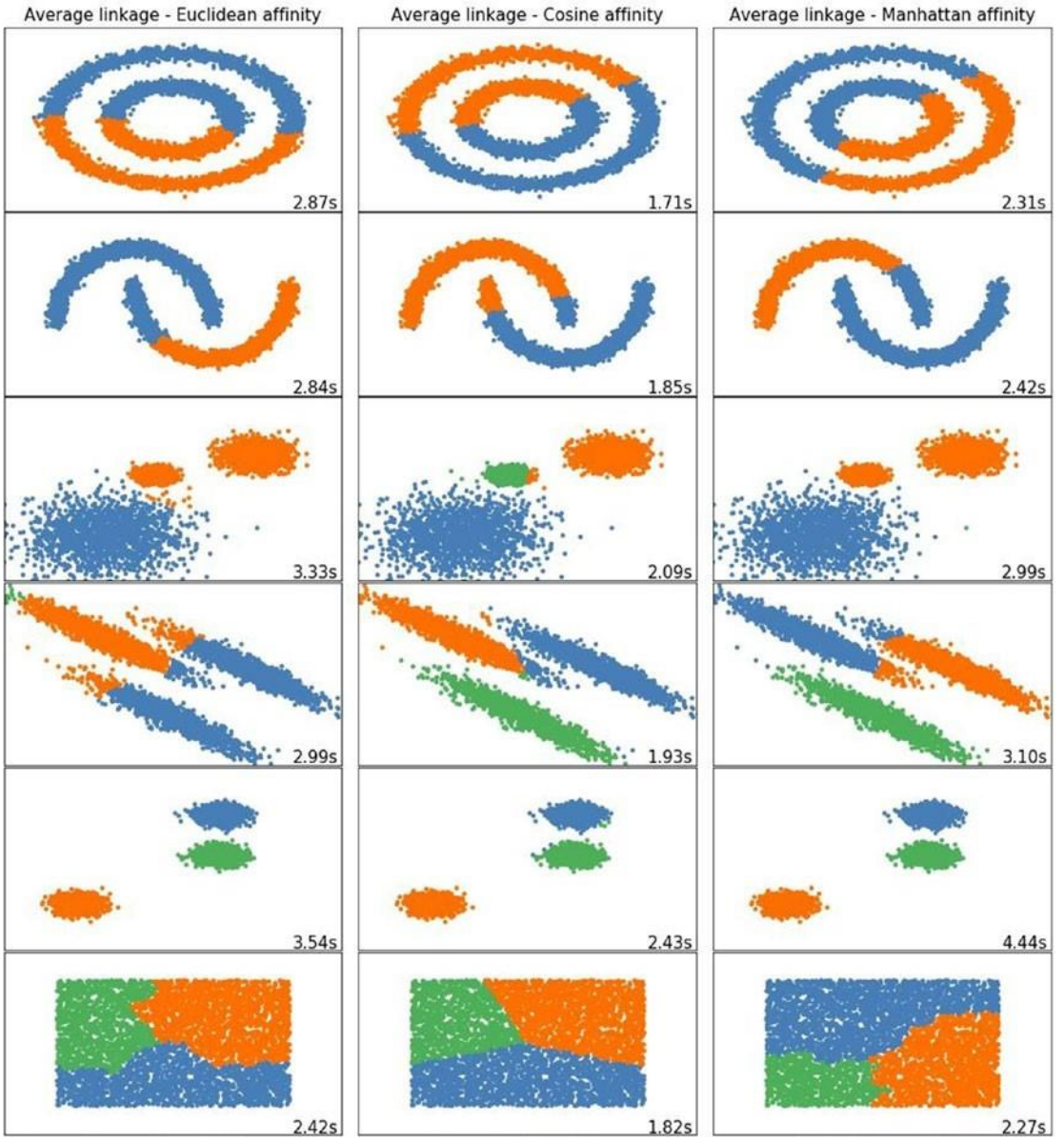


***图 6-20。** 层次 - 相关性（affinity）影响（linkage 选择 average）*

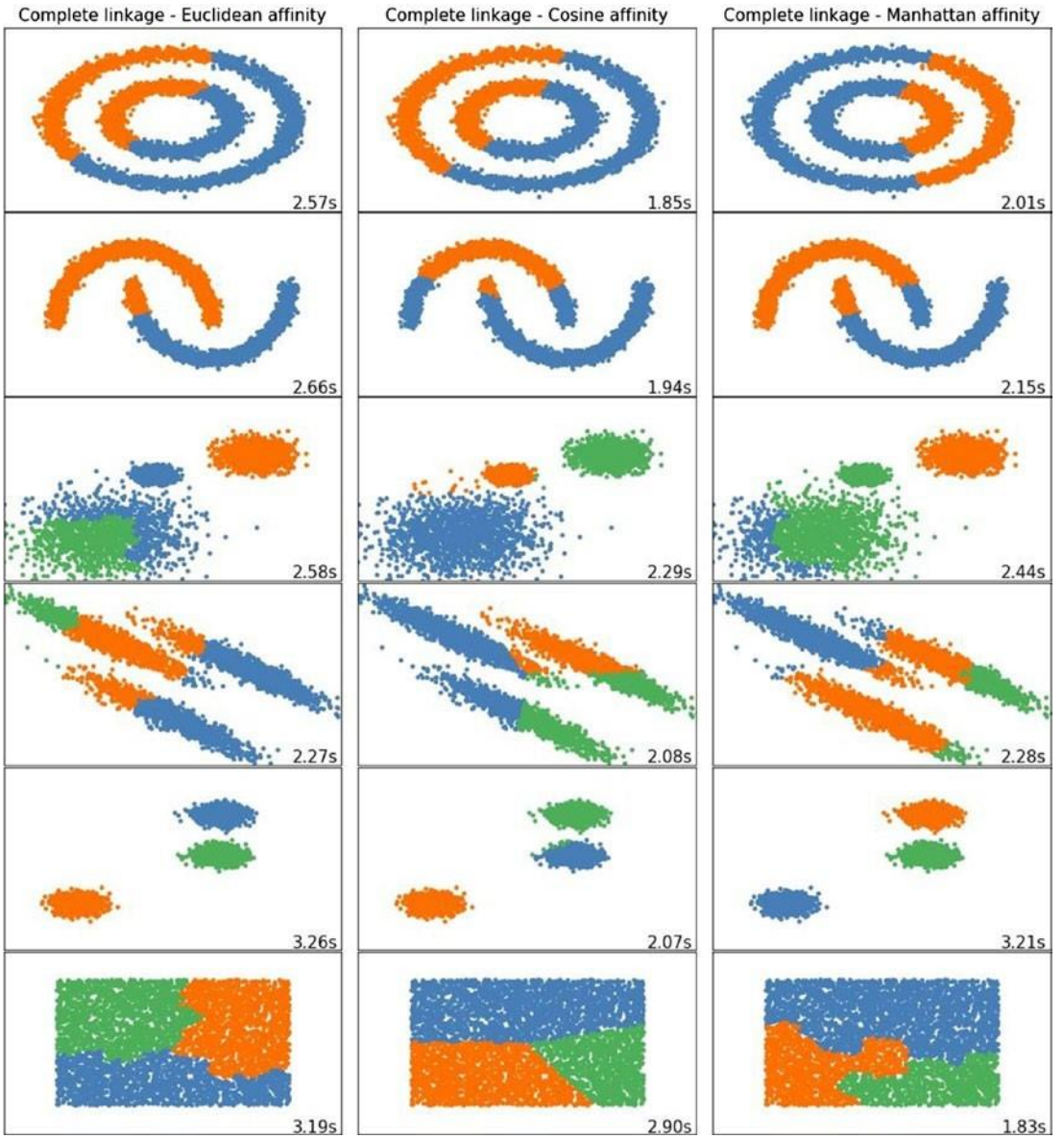


图6-21。 *层次 - 相关性影响（linkage 选择 complete）*

结果表明，相关性对聚类算法有明显影响。

建议 选择精确的参数和调整使用的算法的能力是普通人和数据科学家之间最大的区分。我建议你对于每种算法在不同的数据集上训练，并且

测试大量不同的模式来了解每个算法的参数对机器学习过程的结果的影响。

您正在取得重大进步，可以成功执行层次聚类的解决方案。

在第 11、12、13、14 和 15 章中，我将指导您学习几个补充示例，说明如何使用层次聚类模型的工业化版本来解决实际问题。

现在，我将指导您学习一项重要的技术，该技术对实际解决方案非常有用。

异常检测

异常检测是一种用于识别不符合预期操作（称为异常值）的异常模式的技术。简单地说，它们与正常数据不同。

异常通常可分为以下类型。

点异常

如果数据与其余数据点的距离太远，则这个数据点是异常的。

业务用例（Business use case:）：

根据"花费金额"检测信用卡欺诈。

上下文异常

这种异常依赖于上下文。这种类型的异常在时间序列数据中很常见。

业务用例：

在节日期间每天花费 150.00 美元买食物是正常的， 但除此之外则可能很奇怪。

突然在一个你从来没有去过的地方花钱，或在同一段时间内在不同的两个地方消费，这种不切实际的行为是上下文异常。

集体异常

由元素的多个相关实例组成的异常。

业务用例：

有人尝试非法地将数据从远程计算机复制到本地主机，这种异常将被标记为潜在的网络攻击。

打开代码：Chapter-006-012-Anomaly-Detection-01.ipynb

我们将调查你在哪里买咖啡。如果有人使用您的卡在这些地点之外购买咖啡，这可能是潜在的欺诈行为。

我将向您展示如何使用以下检测过程：

- Robust covariance（鲁棒性的协方差估计）
- One-Class SVM（单分类 SVM）
- Isolation Forest（孤立森林）
- Local Outlier Factor（局部异常因子）

执行代码。

蓝色点不是您的交易。我建议我们联系你的银行，因为你一定是两张卡之一的所有者但是我想你有一张克隆卡。

结果如图 6-22 所示。

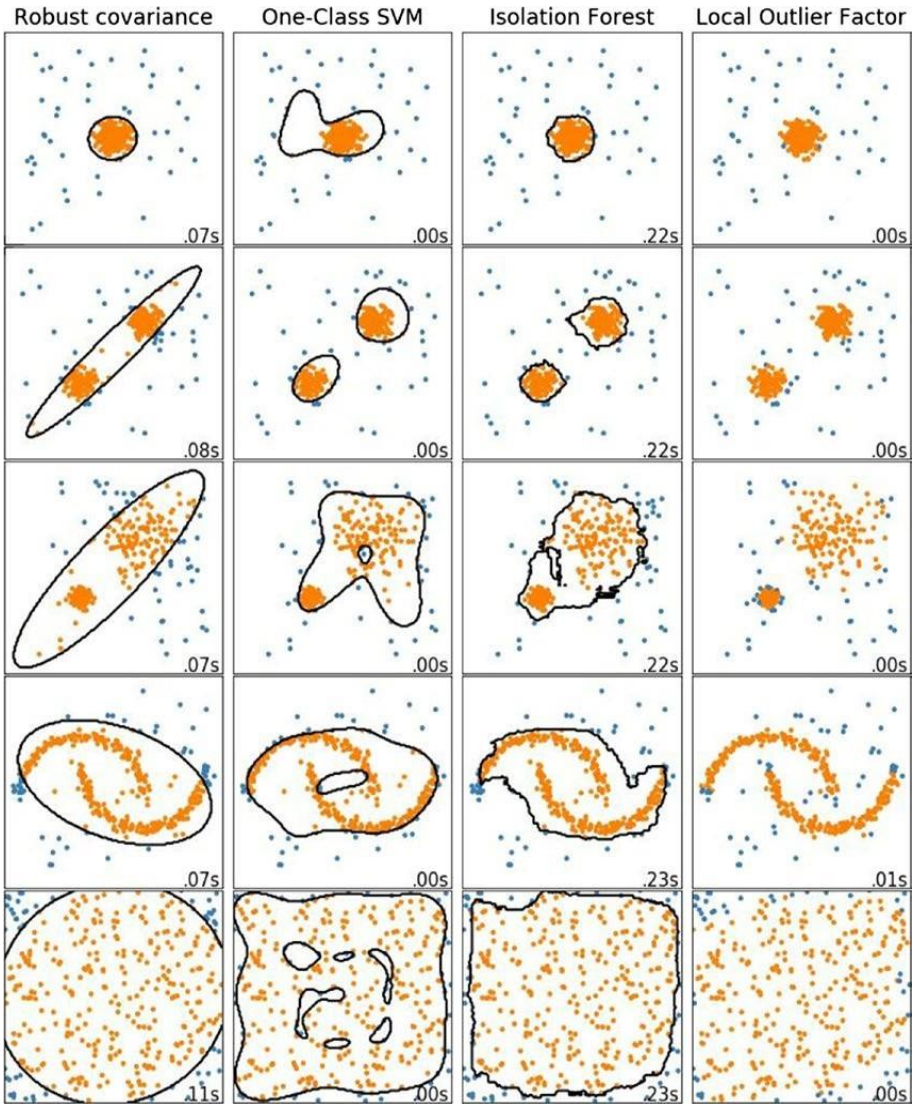


图6-22。 结果

做得好；您已经完成了无监督机器学习的三个部分中的第一个。

在第 11、12、13、14 和 15 章中，我将介绍几个补充示例，说明如何使用工业化版本的异常检测模型解决实际问题。

接下来是什么？

现在，您已经成功地完成了无监督机器学习的第一部分。

我建议你练习新学习的这些算法。

无监督机器学习的第二部分将介绍可用于处理数据洪流的高级算法。