1 线性回归

以氧化物和AH构建模型，并由一组氧化物值估计AH

特征：

0 Date (DD/MM/YYYY)   
1 Time (HH.MM.SS)   
2 True hourly averaged concentration CO in mg/m^3 (reference analyzer)   
3 PT08.S1 (tin oxide) hourly averaged sensor response (nominally CO targeted)   
4 True hourly averaged overall Non Metanic HydroCarbons concentration in microg/m^3 (reference analyzer)   
5 True hourly averaged Benzene concentration in microg/m^3 (reference analyzer)   
6 PT08.S2 (titania) hourly averaged sensor response (nominally NMHC targeted)   
7 True hourly averaged NOx concentration in ppb (reference analyzer)   
8 PT08.S3 (tungsten oxide) hourly averaged sensor response (nominally NOx targeted)   
9 True hourly averaged NO2 concentration in microg/m^3 (reference analyzer)   
10 PT08.S4 (tungsten oxide) hourly averaged sensor response (nominally NO2 targeted)   
11 PT08.S5 (indium oxide) hourly averaged sensor response (nominally O3 targeted)

响应：

14 AH Absolute Humidity

先异常值处理：

#Z-Score去异常值

#去-200缺失值

先试着用seaborn包。网上说这个包的数据可视化效果比较好看。其实seaborn也应该属于matplotlib的内部包。只是需要再次的单独安装。

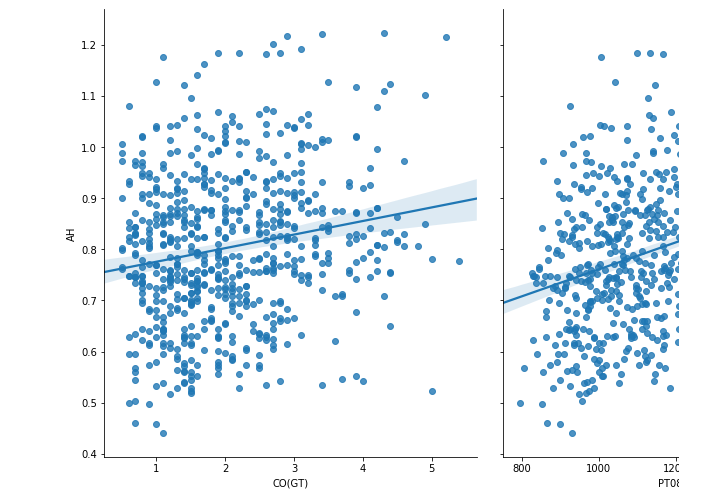
seaborn的pairplot函数绘制X的每一维度和对应Y的散点图。通过设置size和aspect参数来调节显示的大小和比例。可以从图中看出，TV特征和销量是有比较强的线性关系的，而Radio和Sales线性关系弱一些，Newspaper和Sales线性关系更弱。通过加入一个参数kind='reg'，seaborn可以添加一条最佳拟合直线和95%的置信带。

(1)、使用pandas来构建X(特征向量)和y(标签列)

scikit-learn要求X是一个特征矩阵，y是一个NumPy向量。

pandas构建在NumPy之上。

因此，X可以是pandas的DataFrame，y可以是pandas的Series，scikit-learn可以理解这种结构。



也可用sklearn的包：

(1)、使用pandas来构建X(特征向量)和y(标签列)

scikit-learn要求X是一个特征矩阵，y是一个NumPy向量。

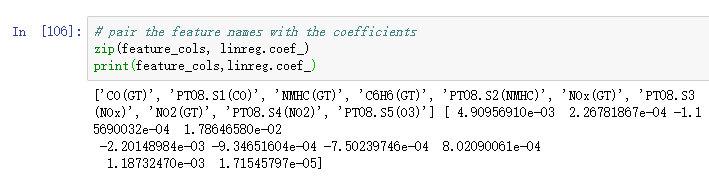
pandas构建在NumPy之上。

因此，X可以是pandas的DataFrame，y可以是pandas的Series，scikit-learn可以理解这种结构。

（2）、构建训练集与测试集

（3）sklearn的线性回归

参数

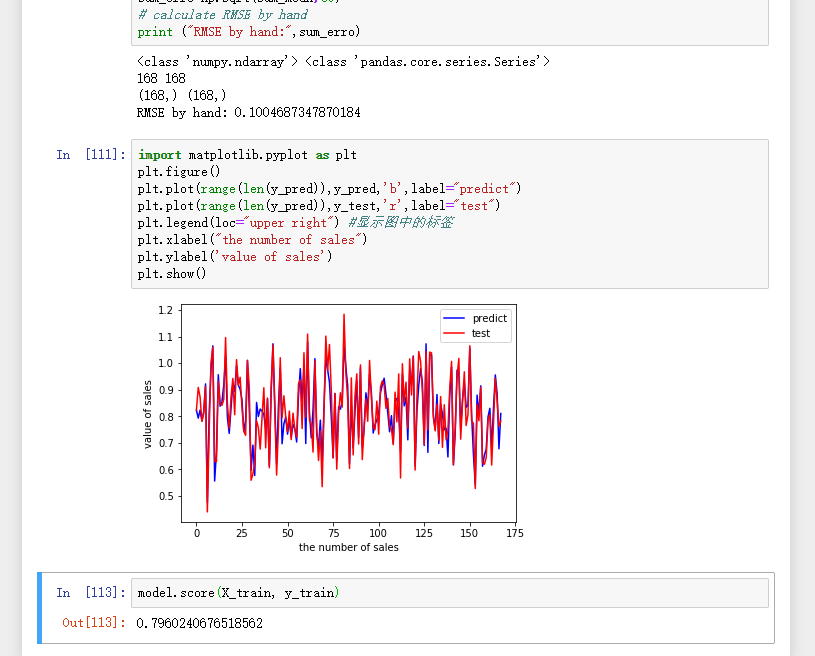


回归问题的评价测度

(1) 评价测度

对于分类问题，评价测度是准确率，但这种方法不适用于回归问题。我们使用针对连续数值的评价测度(evaluation metrics)。

使用均方根误差(Root Mean Squared Error, RMSE)，并作ROC曲线。



2分类

分类：有监督 训练一个模型 让新元素进去时归类

2.1 SVM

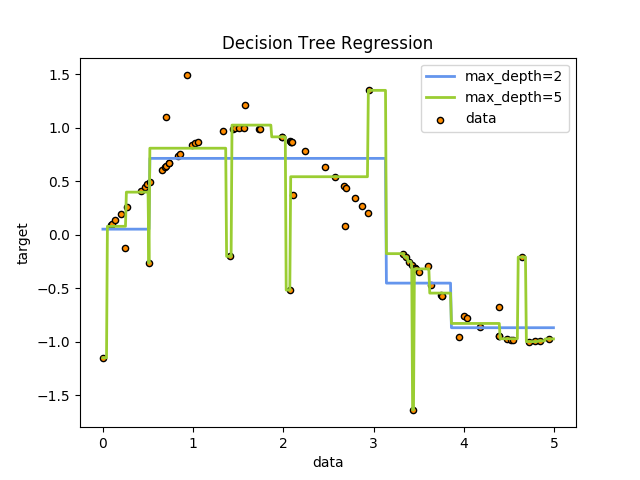
SVC（C-Support Vector Classification）：支持向量分类，基于libsvm实现的（libsvm详情参考 或者百科），数据拟合的时间复杂度是数据样本的二次方，这使得他很难扩展到10000个数据集，当输入是多类别时（SVM最初是处理二分类问题的），通过一对一的方案解决。

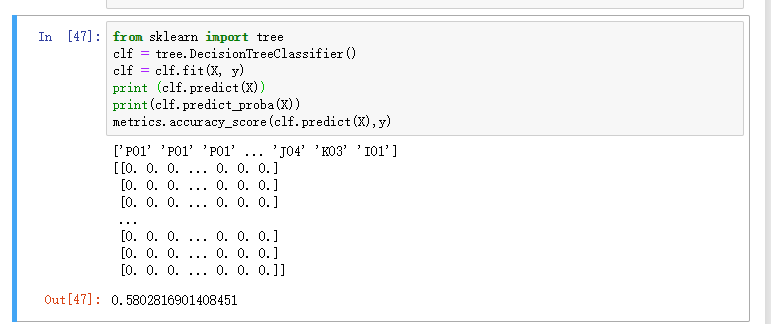


2.2决策树

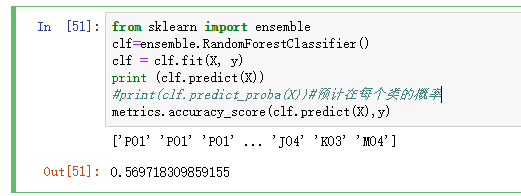
Decision Trees (DTs) 是一种用来 [classification](http://sklearn.apachecn.org/#/docs/11?id=tree-classification) 和 [regression](http://sklearn.apachecn.org/#/docs/11?id=tree-regression) 的无参监督学习方法。其目的是创建一种模型从数据特征中学习简单的决策规则来预测一个目标变量的值。

例如，在下面的图片中，决策树通过if-then-else的决策规则来学习数据从而估测数一个正弦图像。决策树越深入，决策规则就越复杂并且对数据的拟合越好。

[](http://sklearn.apachecn.org/#/../auto_examples/tree/plot_tree_regression.html)



2.3随机森林



3聚类  
聚类：无监督 归类

3.1

DBScan

The [DBSCAN](http://sklearn.apachecn.org/#/generated/sklearn.cluster.DBSCAN.html?id=sklearn.cluster.dbscan) 算法将聚类视为被低密度区域分隔的高密度区域。由于这个相当普遍的观点， DBSCAN发现的聚类可以是任何形状的，与假设聚类是 convex shaped 的 K-means 相反。 DBSCAN 的核心概念是 core samples, 是指位于高密度区域的样本。 因此一个聚类是一组核心样本，每个核心样本彼此靠近（通过一定距离度量测量） 和一组接近核心样本的非核心样本（但本身不是核心样本）。算法中的两个参数, min\_samples 和 eps,正式的定义了我们所说的 [\*](http://sklearn.apachecn.org/#/docs/22?id=id16)dense\*（稠密）。较高的 min\_samples 或者

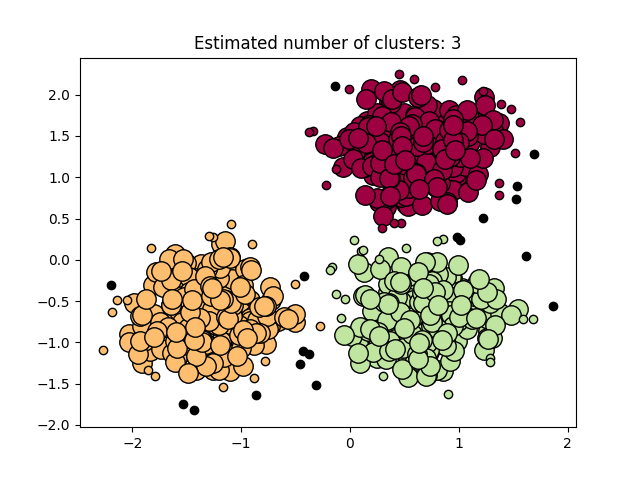
较低的 [``](http://sklearn.apachecn.org/#/docs/22?id=id18)eps``表示形成聚类所需的较高密度。

更正式的,我们定义核心样本是指数据集中的一个样本，存在 min\_samples 个其他样本在 eps 距离范围内，这些样本被定为为核心样本的邻居 neighbors 。这告诉我们核心样本在向量空间稠密的区域。 一个聚类是一个核心样本的集合，可以通过递归来构建，选取一个核心样本，查找它所有的 neighbors （邻居样本） 中的核心样本，然后查找 their（新获取的核心样本）的 neighbors （邻居样本）中的核心样本，递归这个过程。 聚类中还具有一组非核心样本，它们是集群中核心样本的邻居的样本，但本身并不是核心样本。 显然，这些样本位于聚类的边缘。

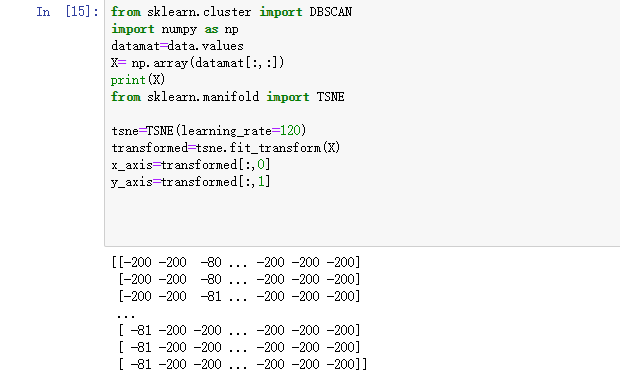
根据定义，任何核心样本都是聚类的一部分，任何不是核心样本并且和任意一个核心样本距离都大于

eps 的样本将被视为异常值。

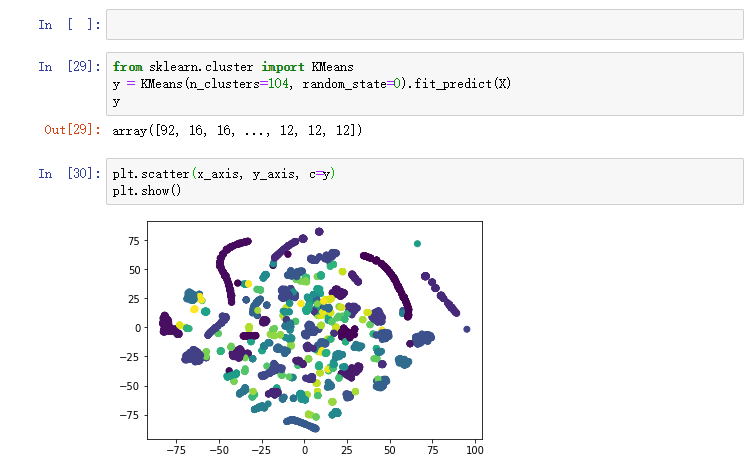
在下图中，颜色表示聚类成员属性，大圆圈表示算法发现的核心样本。 较小的圈子是仍然是群集的 一部分的非核心样本。 此外，异常值由下面的黑点表示。

[](http://sklearn.apachecn.org/#/../auto_examples/cluster/plot_dbscan.html)

先用TSNE降维。



再实现



3.2Kmeans

[KMeans](http://sklearn.apachecn.org/#/generated/sklearn.cluster.KMeans.html?id=sklearn.cluster.kmeans) 算法通过试图分离 n groups of equal variance（n 个相等方差组）的样本来聚集数据，minimizing （最小化）称为 [inertia](http://sklearn.apachecn.org/#/inertia) 或者 within-cluster sum-of-squares （簇内和平方）的 criterion （标准）。 该算法需要指定 number of clusters （簇的数量）。它可以很好地 scales （扩展）到 large number of samples（大量样本），并已经被广泛应用于许多不同领域的应用领域。

k-means 算法将一组 N 样本 X 划分成 K 不相交的 clusters （簇） C, 每个都用 cluster （该簇）中的样本的均值 \mu_j描述。 这个 means （均值）通常被称为 cluster（簇）的 “centroids（质心）”; 注意，它们一般不是从 X 中挑选出的点，虽然它们是处在同一个 space（空间）。 K-means（K-均值）算法旨在选择最小化 inertia（惯性） 或 within-cluster sum of squared（簇内和的平方和）的标准的 centroids（质心）:

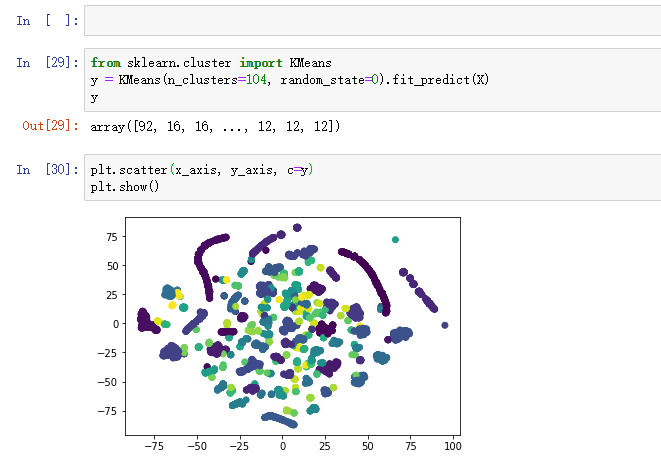
\sum_{i=0}^{n}\min_{\mu_j \in C}(||x_j - \mu_i||^2)

Inertia（惯性）, 或 the within-cluster sum of squares（簇内和平方差） criterion（标准）,可以被认为是 internally coherent clusters （内部想干聚类）的 measure （度量）。 它有各种缺点:

Inertia（惯性）假设 clusters （簇）是 convex（凸）的和 isotropic （各项同性），这并不是总是这样。它对 elongated clusters （细长的簇）或具有不规则形状的 manifolds 反应不佳。

Inertia（惯性）不是一个 normalized metric（归一化度量）: 我们只知道 lower values （较低的值）是更好的，并且 零 是最优的。但是在 very high-dimensional spaces （非常高维的空间）中，欧几里得距离往往会变得 inflated （膨胀）（这就是所谓的 “curse of dimensionality （维度诅咒/维度惩罚）”）。在 k-means 聚类之前运行诸如 [PCA](http://sklearn.apachecn.org/#/PCA)之类的 dimensionality reduction algorithm （降维算法）可以减轻这个问题并加快计算速度。

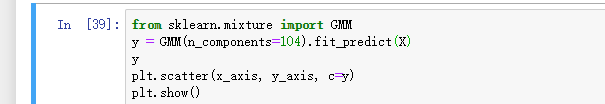
K-means 通常被称为 Lloyd’s algorithm（劳埃德算法）。在基本术语中，算法有三个步骤。、 第一步是选择 initial centroids （初始质心），最基本的方法是从 X 数据集中选择 k 个样本。初始化完成后，K-means 由两个其他步骤之间的循环组成。 第一步将每个样本分配到其 nearest centroid （最近的质心）。第二步通过取分配给每个先前质心的所有样本的平均值来创建新的质心。计算旧的和新的质心之间的差异，并且算法重复这些最后的两个步骤，直到该值小于阈值。换句话说，算法重复这个步骤，直到质心不再显著移动。

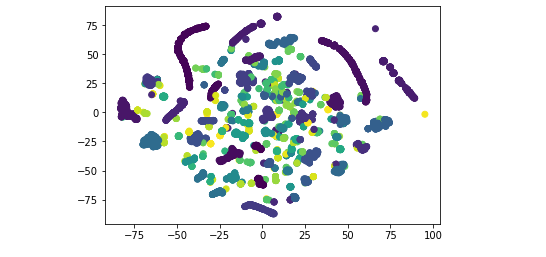


3.3GMM

使用各种GMM分类器绘制训练预测标签，并给出测试数据。

将GMMs与球面、对角、全方差和带结协方差矩阵按性能递增顺序进行比较。尽管人们期望全协方差在一般情况下表现最好，但它很容易对小数据集进行过度拟合，并且不能很好地推广到保留的测试数据。





3.4层次聚类

Hierarchical clustering 是一个常用的聚类算法，它通过不断的合并或者分割来构建聚类。 聚类的层次被表示成树（或者 dendrogram（树形图））。树根是拥有所有样本的唯一聚类，叶子是仅有一个样本的聚类。

The [AgglomerativeClustering](http://sklearn.apachecn.org/" \l "/generated/sklearn.cluster.AgglomerativeClustering.html?id=sklearn.cluster.agglomerativeclustering" \o "sklearn.cluster.AgglomerativeClustering) 使用自下而上的方法进行层次聚类:开始是每一个对象是一个聚类， 并且聚类别相继合并在一起。 linkage criteria 确定用于合并的策略的度量:

Ward 最小化所有聚类内的平方差总和。这是一种 variance-minimizing （方差最小化）的优化方向， 这是与k-means 的目标函数相似的优化方法，但是用 agglomerative hierarchical（聚类分层）的方法处理。

Maximum 或 complete linkage 最小化聚类对两个样本之间的最大距离。

Average linkage 最小化聚类两个聚类中样本距离的平均值。

[AgglomerativeClustering](http://sklearn.apachecn.org/#/generated/sklearn.cluster.AgglomerativeClustering.html?id=sklearn.cluster.agglomerativeclustering) 在于连接矩阵联合使用时，也可以扩大到大量的样本，但是 在样本之间没有添加连接约束时，计算代价很大:每一个步骤都要考虑所有可能的合并。

[FeatureAgglomeration](http://sklearn.apachecn.org/#/generated/sklearn.cluster.FeatureAgglomeration.html?id=sklearn.cluster.featureagglomeration)

The [FeatureAgglomeration](http://sklearn.apachecn.org/" \l "/generated/sklearn.cluster.FeatureAgglomeration.html?id=sklearn.cluster.featureagglomeration" \o "sklearn.cluster.FeatureAgglomeration) 使用 agglomerative clustering 将看上去相似的 特征组合在一起，从而减少特征的数量。这是一个降维工具。

. 