第四章 scikit-learn

教学目标

- 了解
- 掌握

目录

scikit-learn简介

scikit-learn

是专门面向机器学习的Python开源框架，它实现了各种成熟的算法，并且易于安装与使用。

特点：

- 简单有效的数据挖掘和数据分析工具
- 可供所有人访问，并可在各种环境中重复使用
- 基于NumPy，SciPy和matplotlib构建
- 开源，商业上可用 - BSD许可证

相关：

scikit-learn的官方网站：https://scikit-learn.org/stable/index.html

中文文档：http://sklearn.apachecn.org/#/

安装方式：Anaconda自动带

scikit-learn简介

scikit-learn提供了一些常用的数据集：

数据集名称	调用方法
鸢尾花数据集	load_iris()
波士顿房价数据集	load_boston()
乳腺癌数据集	load_breast_cancer()
酒数据集	load_wine()
……	……

scikit-learn简介

鸢尾花数据集： sklearn.datasets.**load_iris**(*return_X_y=False*)

Iris也称鸢尾花卉数据集，是一类多重变量分析的数据集。通过花萼长度，花萼宽度，花瓣长度，花瓣宽度4个属性预测鸢尾花卉属于（Setosa，Versicolour，Virginica）三个种类中的哪一类。

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

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris

scikit-learn简介

鸢尾花数据集： sklearn.datasets.**load_iris**(*return_X_y=False*)

例，获得样本10、25和50的类别名称

```
from sklearn.datasets import load_iris
data = load_iris()
data.target[[15,25,50]]
# list(data.target_names)
```

scikit-learn简介

波士顿房价数据集： sklearn.datasets.load_boston(*return_X_y=False*)

常被用于解决回归问题。

Samples total	506
Dimensionality	13
Features	real, positive
Targets	real 5. - 50.

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html#sklearn.datasets.load_boston

scikit-learn简介

波士顿房价数据集：

属性信息（按顺序）：	• 按城镇划分的CRIM人均犯罪率 • ZN占地超过25,000平方英尺的住宅用地比例。 • INDUS每个城镇非零售业务英亩的比例 • CHAS查尔斯河虚拟变量（如果束缚河流，则为1；否则为0） • NOX一氧化氮浓度（百万分之几） • RM每个住宅的平均房间数 • 1940年之前建造的自有单位的年龄比例 • DIS与五个波士顿就业中心的加权距离 • RAD高速公路通行能力指数 • 每10,000美元的税全额财产税税率 • PTRATIO按镇划分的师生比例 • B 1000（Bk-0.63）^ 2其中Bk是按城镇划分的黑人比例 • LSTAT人口地位降低百分比 • MEDV自有住房的中位数价值（以1000美元计）

scikit-learn简介

波士顿房价数据集： sklearn.datasets.**load_boston**(*return_X_y=False*)

例，获得该数据集数据的形状

```
from sklearn.datasets import load_boston

boston = load_boston()

print(boston.data.shape)
```

scikit-learn简介

乳腺癌数据集：　　sklearn.datasets. **load_breast_cancer**(*return_X_y=False*)

经典且非常容易的二分类数据集。

Classes	2
Samples per class	212(M),357(B)
Samples total	569
Dimensionality	30
Features	real, positive

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer

scikit-learn简介

乳腺癌数据集：

属性信息:

- 半径（从中心到外围点的距离的平均值）
- 纹理（灰度值的标准偏差）
- 周长
- 区
- 平滑度（半径长度的局部变化）
- 压实度（周长^ 2 /面积-1.0）
- 凹度（轮廓凹部的严重程度）
- 凹点（轮廓的凹入部分的数量）
- 对称
- 分形维数（“海岸线近似”-1）

为每个图像计算这些特征的平均值，标准误差和“最差”或最大（三个最差/最大值的平均值），从而得到30个特征。例如，字段0是平均半径，字段10是半径SE，字段20是最差半径。

scikit-learn简介

红酒数据集：　　sklearn.datasets. **load_wine**(*return_X_y=False*)

这份数据集包含来自3种不同起源的葡萄酒的共178条记录。13个属性是葡萄酒的13种化学成分。通过化学分析可以来推断葡萄酒的起源，所有属性变量都是连续变量。

Classes	3
Samples per class	[59,71,48]
Samples total	178
Dimensionality	13
Features	real, positive

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_wine

scikit-learn简介

红酒数据集：

属性信息:	• 醇 • 苹果酸 • 灰 • 灰的碱度 • 镁 • 总酚 • 类黄酮 • 非类黄酮酚 • 原花青素 • 色彩强度 • 色调 • 稀释酒的OD280 / OD315 • 脯氨酸

目录

主要内容

- 去除唯一属性
- 处理缺失值
- 特征编码
- 数据标准化、正则化
- 特征选择
- 主成分分析

去除唯一属性

bbs_id	bbs_title	bbs_content	bbs_sender	bbs_sendTime	bbs_toTop	bbs_toTop
1	批评不会让孩子学习更努力	<span st	闫强	2013-10-08 20:50:50	0	null
2	abc	abcdef	闫强	2013-10-08 20:52:33	0	null
3	18种做法毁掉孩子的自信	1.让孩子	闫强	2013-10-08 20:55:09	0	null
4	4个小技巧提升宝宝免疫力	<p>	闫强	2013-10-08 21:10:50	0	null
5	早期教育的重要性	1、早期教育促进	郭瑞鹏	2013-10-08 21:36:13	0	null
6	最新宝宝搞笑语录	1、囧囧是清华大学	qw	2013-10-08 21:47:35	0	null
7	最新宝宝搞笑语录2	<span style="co	qw	2013-10-08 21:49:03	0	null
8	奇思妙想	<span style="co	qw	2013-10-08 21:51:16	0	null

处理缺失值

缺失值处理方法：

- 直接使用含有缺失值的特征
- 删除含有缺失值的特征
- 缺失值补全

处理缺失值-删除特征

最简单的办法就是删除含有缺失值的特征。

给定数据集　$D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中　$\overrightarrow{x_i} = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)})^T$，$i = 1, 2, \dots, N$

假设　$x^{(t)}$属性含有缺失值，则删除该属性特征有：

$\hat{\vec{x}}_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(t-1)}, x_i^{(t+1)}, \dots, x_i^{(d)})^T$，$i = 1, 2, \dots, N$

处理缺失值-补全

1. 均值插补
2. 用同类均值插补
3. 建模预测
4. 高维映射
5. 多重插补
6. 极大似然估计
7. 压缩感知及矩阵补全

处理缺失值-补全

1. 均值插补
2. 用同类均值插补
3. 建模预测
4. 高维映射
5. 多重插补
6. 极大似然估计
7. 压缩感知及矩阵补全

处理缺失值-补全（均值插补）

属性值
- 连续值 ---用有效值的平均值来插补
- 非连续值 ---用有效值的众数来插补

处理缺失值-补全

1. 均值插补
2. 用同类均值插补
3. 建模预测
4. 高维映射
5. 多重插补
6. 极大似然估计
7. 压缩感知及矩阵补全

处理缺失值-补全（用同类均值插补）

思想：首先将样本进行分类，然后以该类中样本的均值来插补。

给定数据集　D = $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中　$\overrightarrow{x_i} = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)})^T$ ，$i = 1, 2, \dots, N$

假设　$x^{(t)}$属性含有缺失值

处理缺失值-补全（用同类均值插补）

思想：首先将样本进行分类，然后以该类中样本的均值来插补。

将在 $x^{(t)}$ 上有缺失值的数据划分到数据集 $\mathrm{D_u}$

$$D_u = \{ (\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_l, y_l), \}$$

将在 $x^{(t)}$ 上有有效值的数据划分到数据集 $\mathrm{D_l}$

$$D_l = \{ (\vec{x}_{l+1}, y_{l+1}), (\vec{x}_{l+2}, y_{l+2}), \dots, (\vec{x}_N, y_N), \}$$

处理缺失值-补全（用同类均值插补）

思想：首先将样本进行分类，然后以该类中样本的均值来插补。

利用层次聚类对D_l进行聚类，设聚类结果为K个簇C_1，C_2，…，C_K

K个簇在 $x^{(t)}$ 上的均值μ_1，μ_2，…，μ_k。

对于$\vec{\mathrm{x}}_\mathrm{i} \in D_\mathrm{u}$，先对其进行聚类预测，设它被判定为属于某个簇$\mathrm{C}_\mathrm{k}$，（$1 \leq \mathrm{k} \leq \mathrm{K}$），则有$\hat{\mathrm{x}}_\mathrm{i}^{(t)} = \mu_\mathrm{k}$

处理缺失值-补全

1. 均值插补
2. 用同类均值插补
3. 建模预测
4. 高维映射
5. 多重插补
6. 极大似然估计
7. 压缩感知及矩阵补全

处理缺失值-补全（建模预测）

思想：将缺失的属性作为预测的目标来预测。

给定数据集　D = $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_{\mathrm{N}}, y_{\mathrm{N}})\}$

其中　$\overrightarrow{x_i} = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)})^{\mathrm{T}}$ ，$i = 1, 2, \dots, N$

假设　$x^{(t)}$属性含有缺失值，且假设：

（$\vec{x}_1$，　$\vec{x}_2$，　…，　$\vec{x}_{\mathrm{N1}}$）在$x^{(t)}$属性上含有有效值

（$\vec{x}_{\mathrm{N1+1}}$，　$\vec{x}_{\mathrm{N1+2}}$，　…，　$\vec{x}_{\mathrm{N}}$）在$x^{(t)}$属性上为缺失值

处理缺失值-补全（建模预测）

思想：将缺失的属性作为预测的目标来预测。

构建新的数据集： $\mathrm{D_t} = \{(\hat{\vec{x}}_1, x_1^{(\mathrm{t})}), (\hat{\vec{x}}_2, x_2^{(\mathrm{t})}), \dots, (\hat{\vec{x}}_{\mathrm{N1}}, x_{\mathrm{N1}}^{(\mathrm{t})})\}$

构建待预测数据集为： $\mathrm{D_p} = \{\hat{\vec{x}}_{\mathrm{N1+1}}, \hat{\vec{x}}_{\mathrm{N1+2}}, \dots, \hat{\vec{x}}_{\mathrm{N}}\}$ 其中

$\hat{\vec{\mathrm{x}}}_{\mathrm{i}} = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(\mathrm{t-1})}, x_i^{(t+1)}, \dots, x_i^{(d)})^{\mathrm{T}}$ ， $i = 1, 2, \dots, N$

处理缺失值-补全

1. 均值插补
2. 用同类均值插补
3. 建模预测
4. 高维映射
5. 多重插补
6. 极大似然估计
7. 压缩感知及矩阵补全

处理缺失值-补全（高维映射）

思想：将属性映射到高维空间。

给定数据集　D = $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中　$\overrightarrow{x_i} = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)})^T$ ，$i = 1, 2, \dots, N$

假设　$x^{(t)}$属性的取值为离散值 $\{a_{t,1}, a_{t,2}, \dots, a_{t,k}\}$ 且含有缺失值

将该属性扩展为 K+1 个属性$\left(x^{(t,1)}, x^{(t,2)}, \dots, x^{(t,k)}, x^{(t,k+1)}\right)$

处理缺失值-补全（高维映射）

则：

$\hat{\hat{\mathrm{X}}}_{\mathrm{i}} = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(\mathrm{t}-1)}, x_i^{(t,1)}, x_i^{(t,2)}, \ldots, x_i^{(t,k)}, x_i^{(t,k+1)}, x_i^{(t+1)}, \ldots, x_i^{(d)})^{\mathrm{T}}$，
$i = 1, 2, \ldots, N$

$$x_i^{(t,j)} = \begin{cases} 1, & \text{if } j = K+1 \text{ and } x_i^{(t)} \text{ miss} \\ 1, & \text{if } i \le j \le K \text{ and } x_i^{(t)} = \mathrm{a}_{t,j},\ j = 1, 2, \ldots, K+1 \\ 0, & \text{else} \end{cases}$$

处理缺失值-补全

1. 均值插补
2. 用同类均值插补
3. 建模预测
4. 高维映射
5. 多重插补
6. 极大似然估计
7. 压缩感知及矩阵补全

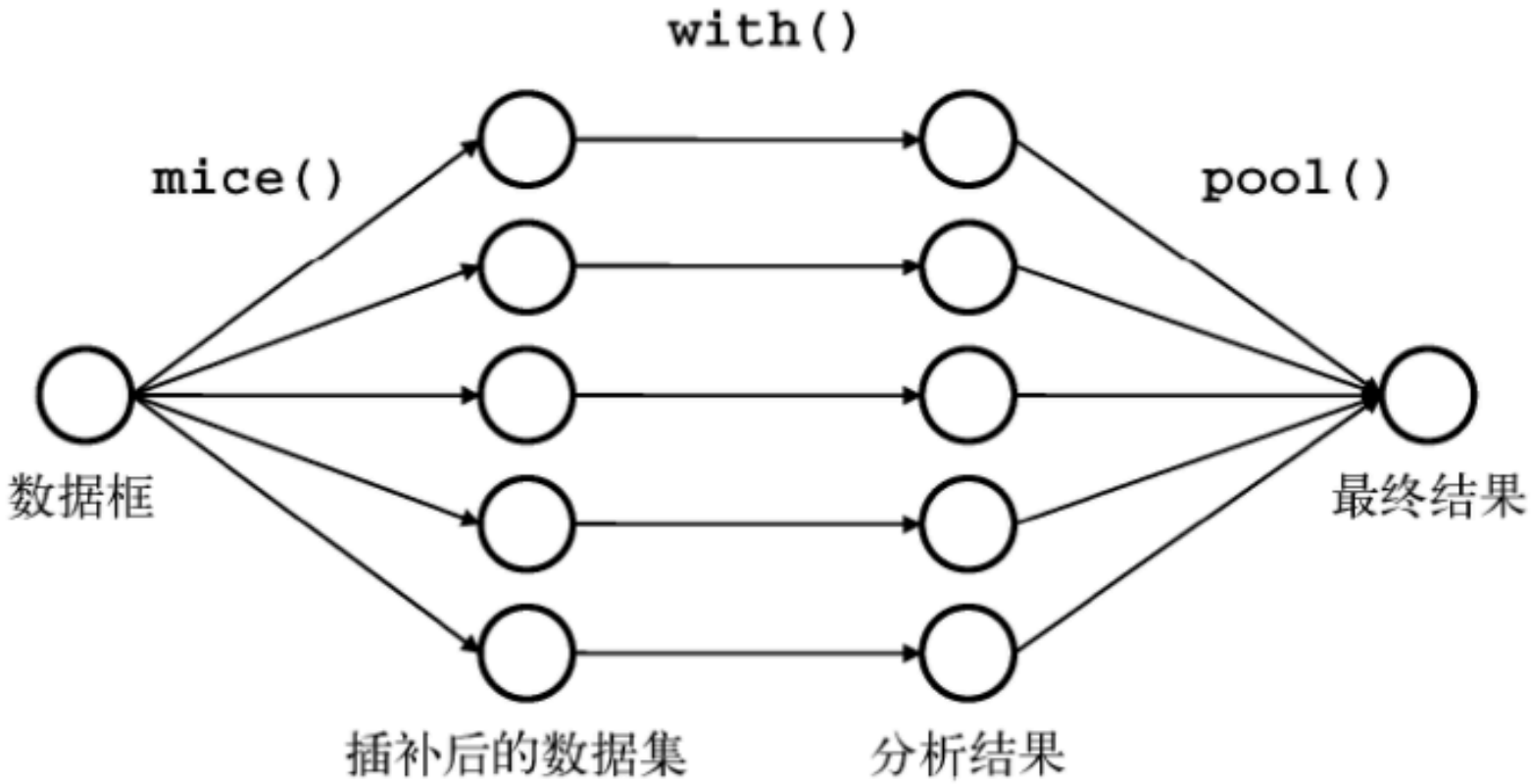
处理缺失值-补全（多重插补）

认为待插补的值是随机的，它的值来自于已观测到的值。

多重插补的方法分为3个步骤：

1. 通过变量之间的关系对缺失数据进行预测，利用蒙特卡洛方法生成多个完整的数据集
2. 在每个完整的数据集上进行训练，得到训练后的模型以及评价函数值
3. 对来自各个完整的数据集的结果，根据评价函数值进行选择，选择评价函数值最大的模型，其对应的差值就是最终的插补值

处理缺失值-补全（多重插补）



处理缺失值-补全

应该怎么选？

主要内容

- 去除唯一属性
- 处理缺失值
- 特征编码
- 数据标准化、正则化
- 特征选择
- 主成分分析

特征编码-特征二元化

特征二元化的过程试讲数值型的属性转换为布尔值的属性。

通常用于假设属性取值分布为伯努利分布的情形。

给定数据集　D = $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中　$\overrightarrow{x_i} = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)})^T$，$i = 1, 2, \dots, N$

对于某个属性　$x^{(j)}$　其取值集合为　$\{x_1^{(j)}, x_2^{(j)}, \dots, x_N^{(j)}\}$

若指定一个阈值ϵ，则:

$$\begin{cases} \text{当}\quad x^{(j)} \geq \epsilon,\ \hat{x}^j = 1 \\ \text{当}\quad x^{(j)} < \epsilon,\ \hat{x}^j = 0 \end{cases}$$

特征编码-独热编码

如有数据：

[男、女]

[中国、美国、英国]

如何处理？

特征编码-独热编码

独热编码即 One-Hot 编码，又称一位有效编码，其方法是使用N位状态寄存器来对N个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候，其中只有一位有效。

即，将该属性编码为一个m元的元组：

$$(v_1, v_2, \dots, v_m) v_i \in \{0,1\}, \qquad i = 1, 2, \dots, m$$

$(v_1, v_2, \dots, v_m)$ 的分量有且仅有一个为1，其余的分量均为0

特征编码-独热编码

[男、女]

男 → (1, 0)

女 → (0, 1)

[中国、美国、英国]

中国 → (1, 0, 0)

美国 → (0, 1, 0)

英国 → (0, 0, 1)

男，中国 → (1, 0, 1, 0, 0)

特征编码-独热编码

独热编码3个特点：

1. 处理非数值属性
2. 扩充特征
3. 编码后属性稀疏，存在大量的零元分量

主要内容

- 去除唯一属性
- 处理缺失值
- 特征编码
- 数据标准化、正则化
- 特征选择
- 主成分分析

数据标准化、正则化-标准化

有些算法要求样本数据具有零均值和单位方差；

样本不同属性具有不同量级时，需要消除量级的影响；

……

数据标准化是将样本的属性缩放到某个指定的范围。

- ✓ min-max标准化
- ✓ z-score标准化

数据标准化、正则化-正则化

数据正则化，是将样本的某个范数缩放到单位1。

对于样本$\vec{x}_i$,先计算其L_p范数：

$$L_p(\vec{x}_i) = \left(\left|x_i^{(1)}\right|^p + \left|x_i^{(2)}\right|^p + \cdots + \left|x_i^{(d)}\right|^p\right)^{1/p}$$

每个属性值除以其L_p范数：

$$\hat{\vec{x}}_i = \left(\frac{x_i^{(1)}}{L_p(\vec{x}_i)}, \frac{x_i^{(2)}}{L_p(\vec{x}_i)}, \cdots, \frac{x_i^{(d)}}{L_p(\vec{x}_i)}\right)^T$$

主要内容

- 去除唯一属性
- 处理缺失值
- 特征编码
- 数据标准化、正则化
- 特征选择
- 主成分分析

特征选择

从给定的特征集合中选出相关特征子集的过程称为特征选择（feature selection）。

- 属性过多会造成维数灾难问题，需要去除不相关特征来降低学习任务的难度。
- 方法：
  1. 过滤式、2. 包裹式、 3. 嵌入式

特征选择-过滤式选择

过滤式方法先对数据集进行特征选择，然后再训练学习器。

以过滤式特征选择方法中的relief方法为例：

给定训练集　　$D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

其中　　$\vec{x}_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)})^T$，$i = 1, 2, \dots, N$，$y_i \in \{-1, 1\}$

对每个样本$\vec{x}_i$，i = 1, 2, …, N：

1. 在$\vec{x}_i$同类样本中寻找其最近邻$\vec{x}_{i,nh}$，称为猜中近邻near-hit。
2. 在$\vec{x}_i$异类样本中寻找其最近邻$\vec{x}_{i,nm}$，称为猜错近邻near-miss。
3. 计算$\vec{\delta}_i = \left(\delta_i^{(1)}, \delta_i^{(2)}, \dots, \delta_i^{(d)}, \right)^T$ 对应于属性j的分量$(j = 1, 2, \dots, d)$。

特征选择-过滤式选择

3. 计算$\vec{\delta}_i = \left(\delta_i^{(1)}, \delta_i^{(2)}, \ldots, \delta_i^{(d)}, \right)^T$ 对应于属性j的分量$(j = 1, 2, \ldots, d)$。

$$\delta_i^{(j)} = -diff\left(x_i^{(j)}, x_{i,nh}^{(j)}\right)^2 + diff\left(x_i^{(j)}, x_{i,nm}^{(j)}\right)^2$$

$diff\left(x_i^{(j)}, x_{i,nh}^{(j)}\right)$为两个样本在属性$j$上的差异值：

如果j是离散的，则

$$diff\left(x_a^{(j)}, x_b^{(j)}\right) = \begin{cases} 0, & if \quad x_a^{(j)} = x_b^{(j)} \\ 1, & else \end{cases}$$

如果j是连续的，则

$$diff\left(x_a^{(j)}, x_b^{(j)}\right) = \left|x_a^{(j)} - x_b^{(j)}\right|$$

特征选择-过滤式选择

4. 计算$\vec{\delta}$为$\vec{\delta}_i$的均值

$$\vec{\delta} = \frac{1}{N}\sum_{i=1}^{N}\vec{\delta}_i$$

5. 根据指定的阈值τ，如果$\delta^{(j)} > \tau$则样本属性j被选中。

特征选择-包裹式选择

包裹式特征选择直接把最终将要使用的学习器的性能作为特征子集的评价准则。

以包裹式特征选择方法中的LVW方法为例：

- 输入
  - 数据集　　D = $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$，　$\vec{x}_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(d)})^T$
  - 特征集　　A = $\{x^{(1)}, x^{(2)}, \dots, x^{(d)}\}$
  - 学习器　　estimator
  - 迭代停止条件　　*T*
- 输出：最优特征子集*A**

特征选择-包裹式选择

- 算法步骤如下：

1. 初始化

   将$\tilde{A}^* = A$,交叉验证法评估学习器estimator的误差err*

2. 迭代，迭代终止条件为迭代次数到达T

   随机产生特征子集A'

   评估学习器estimator的误差err

   若err≤ err^*，且A'的特征数量少于$\tilde{A}^*$ ，则$\tilde{A}^* = A'$； $err^* = err$

3. 最终 $A^* = \tilde{A}^*$

目录

监督学习方法

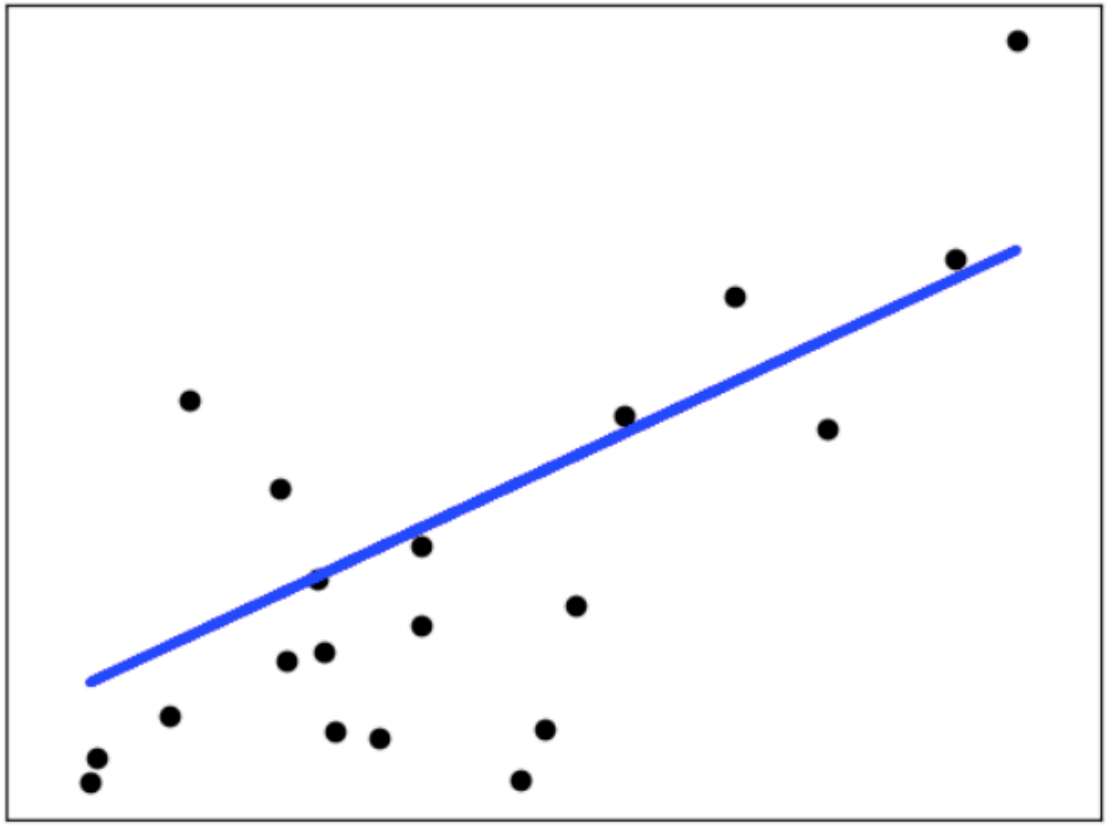
- 回归
  - 线性回归
- 分类
  - 决策树
  - Knn
  - 神经网络（有监督）

监督学习方法

- 回归
  - 线性回归
- 分类
  - 决策树
  - Knn
  - 神经网络（有监督）

监督学习方法-普通最小二乘法

- 线性回归-普通最小二乘法



1. from sklearn import **linear_model**

2. linear_model.**LinearRegression()**

LinearRegression

拟合一个带有系数 $w = (w_1, ..., w_p)$的线性模型，使得数据集实际观测数据和预测数据（估计值）之间的残差平方和最小。

监督学习方法-普通最小二乘法

- 线性回归-普通最小二乘法

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit([[0., 0.],[1., 1.],[2., 2.]],[0, 1, 2])
reg.coef_
```

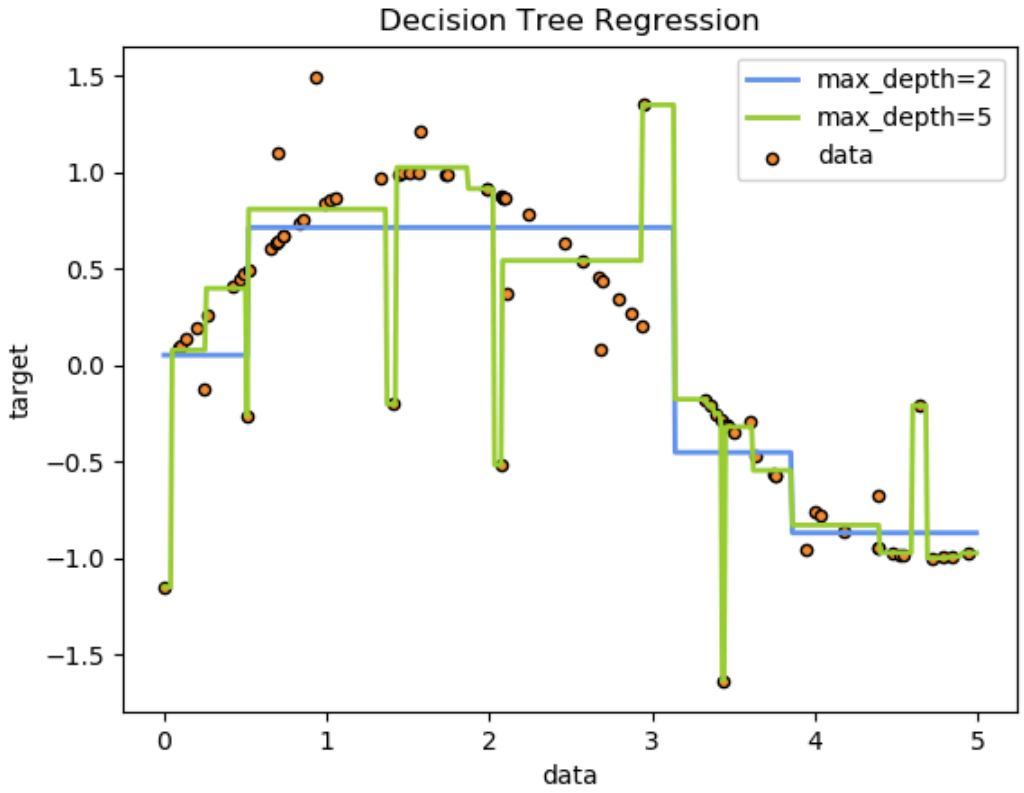
```
array([0.5, 0.5])
```

监督学习方法

- 回归
  - 线性回归
- 分类
  - 决策树
  - Knn
  - 神经网络（有监督）

监督学习方法-决策树

- 分类-决策树
  - 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。
  - 例如下图，通过if-then-else的决策规则来学习数据从而估测出一个正弦图像。



监督学习方法-决策树

- 分类-决策树

DecisionTreeClassifier能够在数据集上执行多分类的类, 采用输入两个数组方式来存放训练样本。

数组X，
存放训练样本

数组Y，
存放训练样本的类标签

```
from sklearn import tree
x = [[0, 0],[1, 1]]
y = [0, 1]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x, y)
```

执行通过之后，可以使用该模型来预测样本类别:

```
clf.predict([[2., 2.]])
```

```
array([1])
```

监督学习方法-决策树

- 分类-决策树

DecisionTreeClassifier能够在数据集上执行多分类的类, 采用输入两个数组方式来存放训练样本。

使用iris数据集构造一个决策树：

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
```

决策树实际使用技巧

监督学习方法

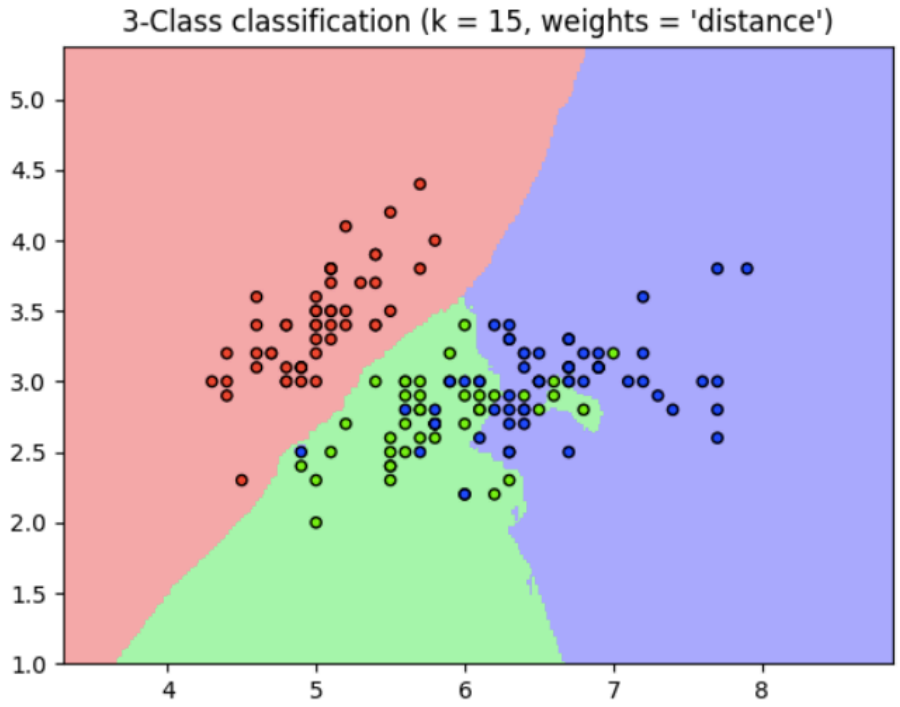
- 回归
  - 线性回归
- 分类
  - 决策树
  - Knn
  - 神经网络（有监督）

监督学习方法-KNN

- 分类-KNN

sklearn.neighbors类能够在数据集上执行多分类。

最近邻分类器：**KNeighborsClassifier**

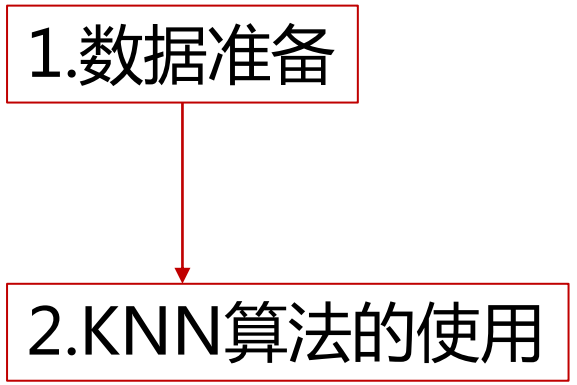


监督学习方法-KNN

- 分类-KNN

sklearn.neighbors类能够在数据集上执行多分类。

基于iris数据集的KNN：



监督学习方法-KNN

- 分类-KNN

sklearn.neighbors类能够在数据集上执行多分类。

基于iris数据集的KNN：

1.数据准备

```
from sklearn import datasets
import numpy as np
iris = datasets.load_iris()
iris_x = iris.data
iris_y = iris.target
```

```
indices = np.random.permutation(len(iris_x))
iris_train_x = iris_x[indices[:-10]]
iris_train_y = iris_y[indices[:-10]]
iris_test_x = iris_x[indices[-10:]]
iris_test_y = iris_y[indices[-10:]]
```

监督学习方法-KNN

- 分类-KNN

sklearn.neighbors类能够在数据集上执行多分类。

基于iris数据集的KNN：

2.KNN算法的使用

```
#使用sklearn制造knn分类器
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
```

```
#训练
knn.fit(iris_train_x,iris_train_y)
```

```
# 测试集来测试
iris_predict_y = knn.predict(iris_test_x)
```

```
#验证结果
iris_test_y - iris_predict_y
```

监督学习方法

- 回归
  - 线性回归
- 分类
  - 决策树
  - Knn
  - 神经网络（有监督）

监督学习方法-神经网络（有监督）

- 分类-神经网络（有监督）

基础概念：多层感知器（MLP）、输入层、输出层、隐藏层、权值矩阵……

MLPClassifier类实现了通过Backpropargation进行训练的多层感知器算法。

```
#训练
from sklearn.neural_network import MLPClassifier
X = [[0., 0.],[1., 1.]]
y = [0, 1]
clf = MLPClassifier()
clf.fit(X, y)
```

```
#预测新样本
clf.predict([[2., 2.],[-1., -2.]])
```

监督学习方法-神经网络（有监督）

- 分类-神经网络（有监督）

基础概念：多层感知器（MLP）、输入层、输出层、隐藏层、权值矩阵……

MLPClassifier类实现了通过Backpropargation进行训练的多层感知器算法。

clf.coefs_ 包含了构建模型的权值矩阵:

```
[coef.shape for coef in clf.coefs_]
```

支持多分类：

```
X = [[0., 0.],[1., 1.]]
y = [[0, 0],[1., 1.]]
```

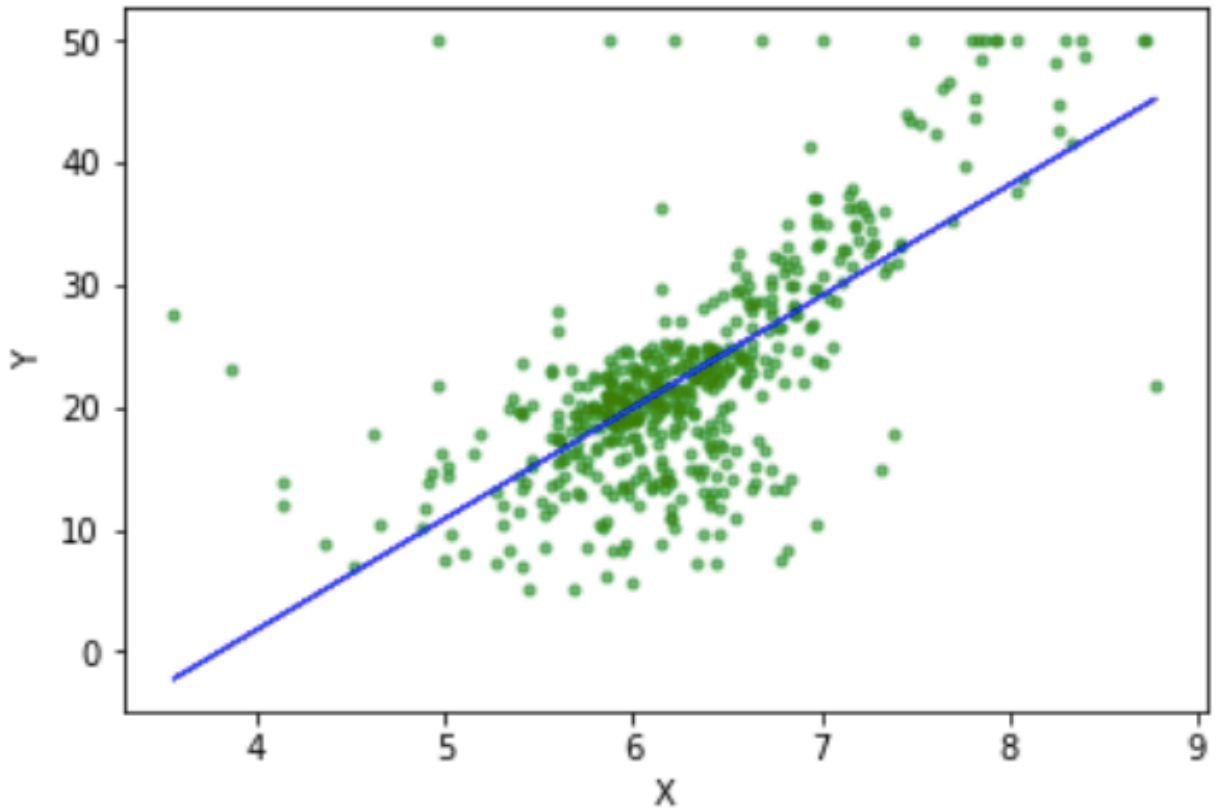
练习1

- 决策树-通过红酒数据集进行拟合查看其预测准确度

```
# 导入需要的算法库和模块
from  sklearn  import  tree
from  sklearn.datasets  import  load_wine
from  sklearn.model_selection  import  train_test_split
# 准备数据
wine = load_wine()
Xtrain, Xtest, Ytrain, Ytest = train_test_split(wine.data,wine.target,test_size=0.3)
#模型载入和拟合
clf = tree.DecisionTreeClassifier()
clf = clf.fit(Xtrain, Ytrain)
score = clf.score(Xtest, Ytest)
```

练习2

- 线性回归-通过波士顿房屋数量对房价进行预测



练习3

- 神经网络（分类）-通过给定数据集进行拟合

预测数据： [0.317029, 14.739025]

```
每层网络层系数矩阵维度:
 [(2, 5), (5, 2), (2, 1)]
预测结果:  [0.]
预测结果概率:
 [[0.99549534 0.00450466]]
```

练习4

- 神经网络（回归）**-通过给定数据集进行拟合**

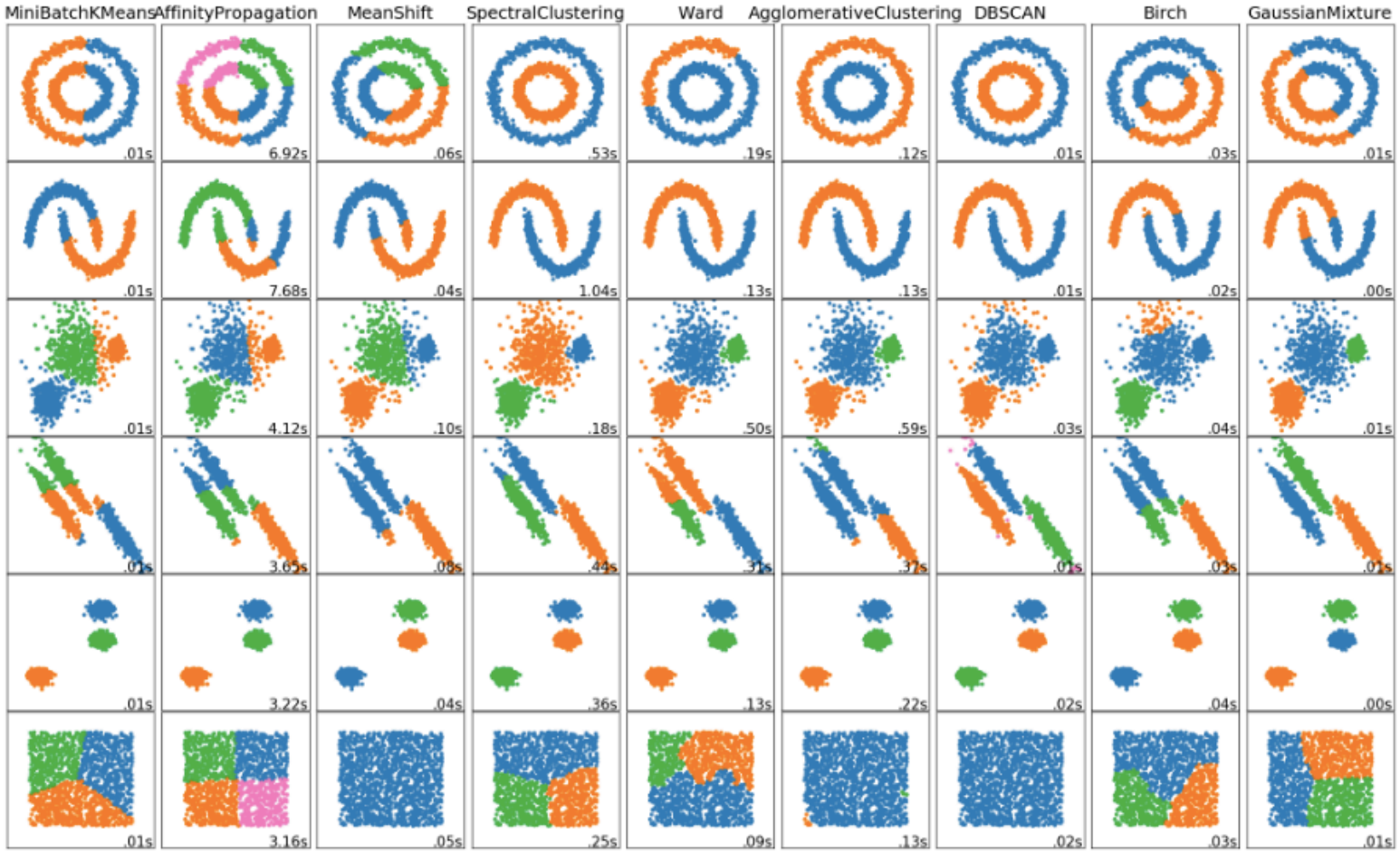
预测数据： [0.317029, 14.739025]

```
预测结果:  [24.27658241]
每层网络层系数矩阵维度:
 [(2, 5), (5, 2), (2, 1)]
```

目录

无监督学习方法

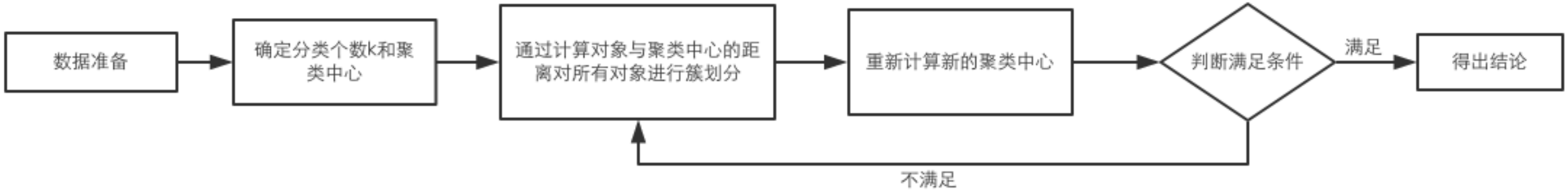
- 聚类
  - KMeans
  - DBSCAN
  - 层次聚类
  - 高斯混合
  - ……



from **sklearn.cluster** import ……

无监督学习方法-KMeans

- 聚类-**KMeans**

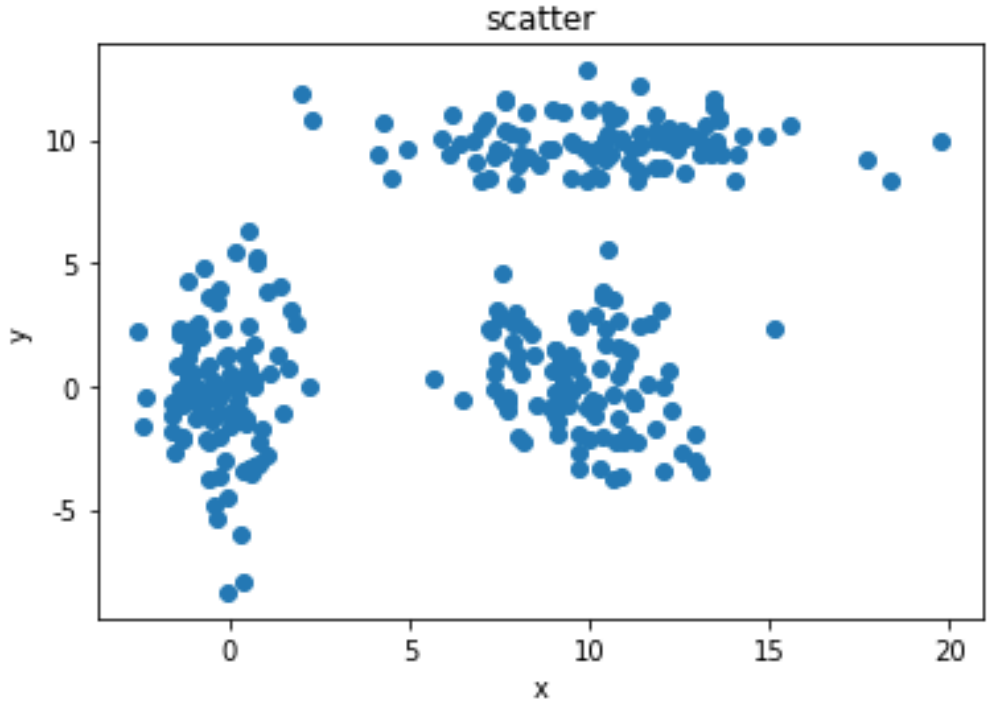


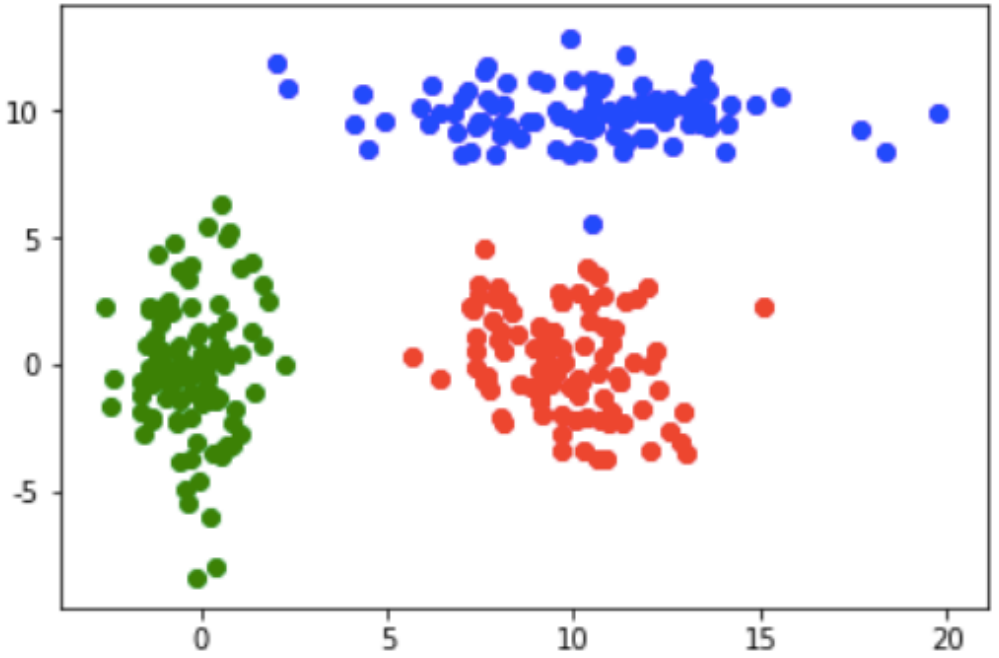
1. form sklearn.cluster import **KMeans**
2. 参数**n_clusters=k**
3. 得到模型

无监督学习方法-KMeans

- 聚类-**KMeans**

KMeans聚类算法对高斯数据集进行聚类





无监督学习方法-KMeans

- 聚类-**KMeans**

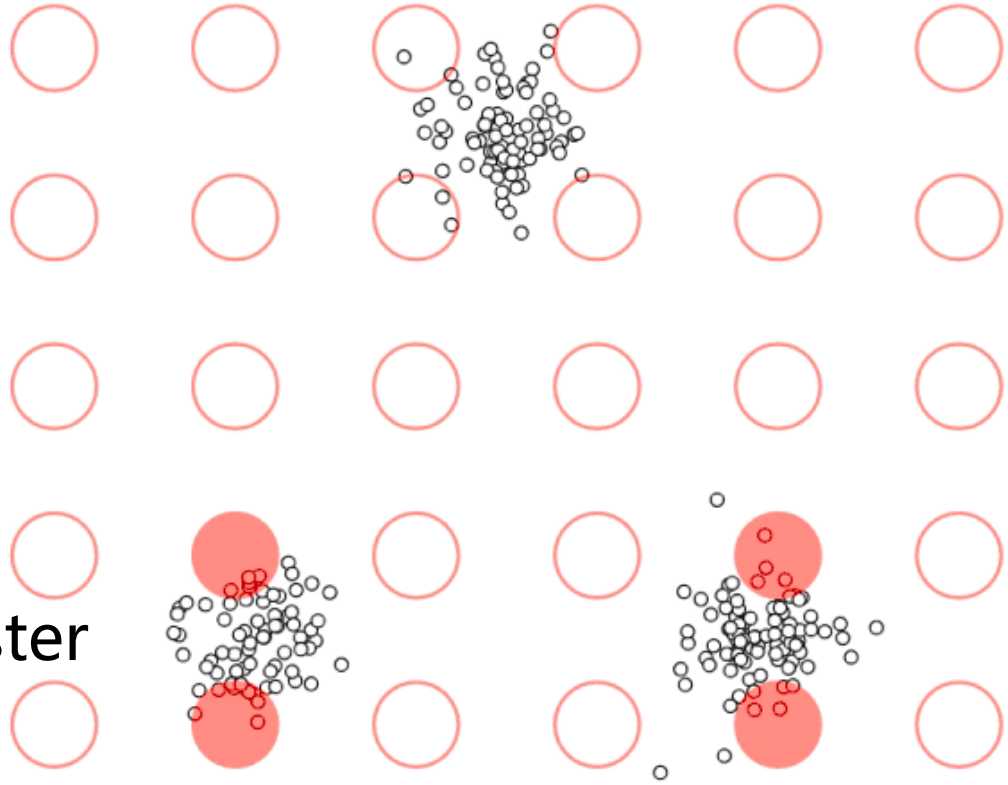
KMeans聚类算法对高斯数据集进行聚类

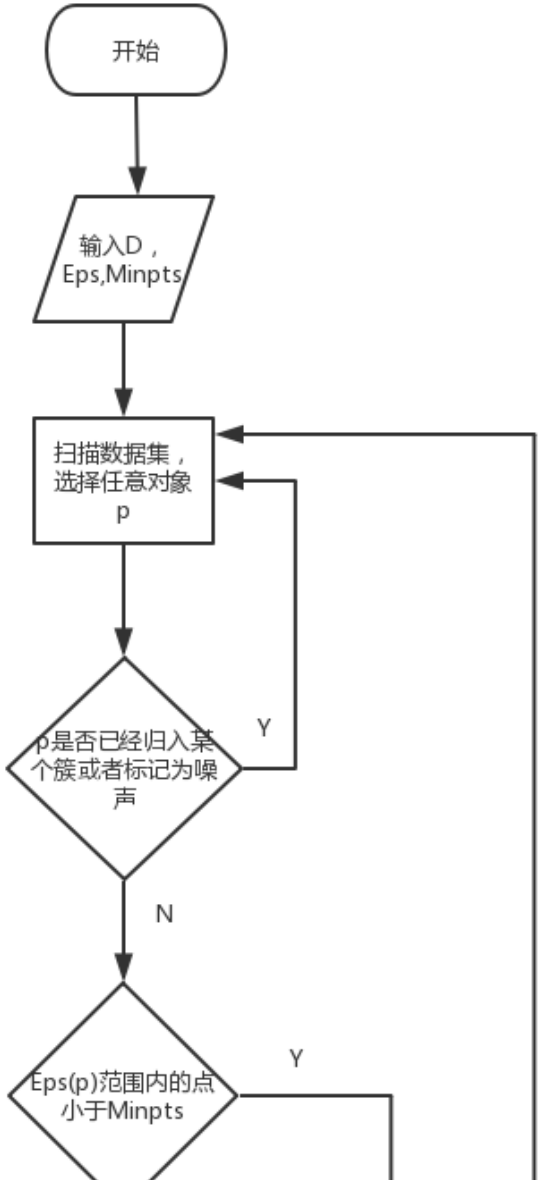
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters = 3)

kmeans.fit(d)
```

```
a = d[kmeans.labels_==0]
b = d[kmeans.labels_==1]
c = d[kmeans.labels_==2]
xa = a[:,0]
ya = a[:,1]
xb = b[:,0]
yb = b[:,1]
xc = c[:,0]
yc = c[:,1]
plt.scatter(xa, ya, c = 'red')
plt.scatter(xb, yb, c = 'green')
plt.scatter(xc, yc, c = 'blue')
plt.show()
```

无监督学习方法-DBSCAN

- 聚类-**DBSCAN**



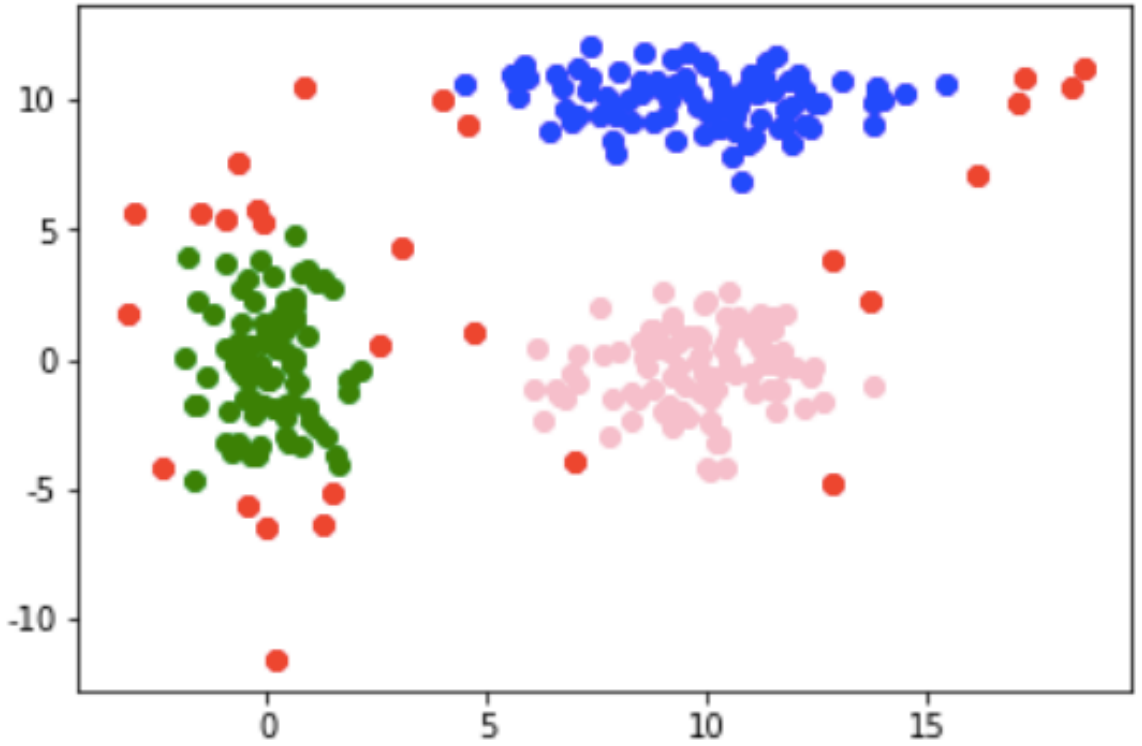


1. form sklearn.cluster import **DBSCAN**
2. 参数 **eps、min_samples**
3. 得到模型

无监督学习方法-DBSCAN

- 聚类-**DBSCAN**

密度聚类算法对高斯数据集进行聚类



无监督学习方法-KMeans

- 聚类-**DBSCAN**

密度聚类算法对高斯数据集进行聚类

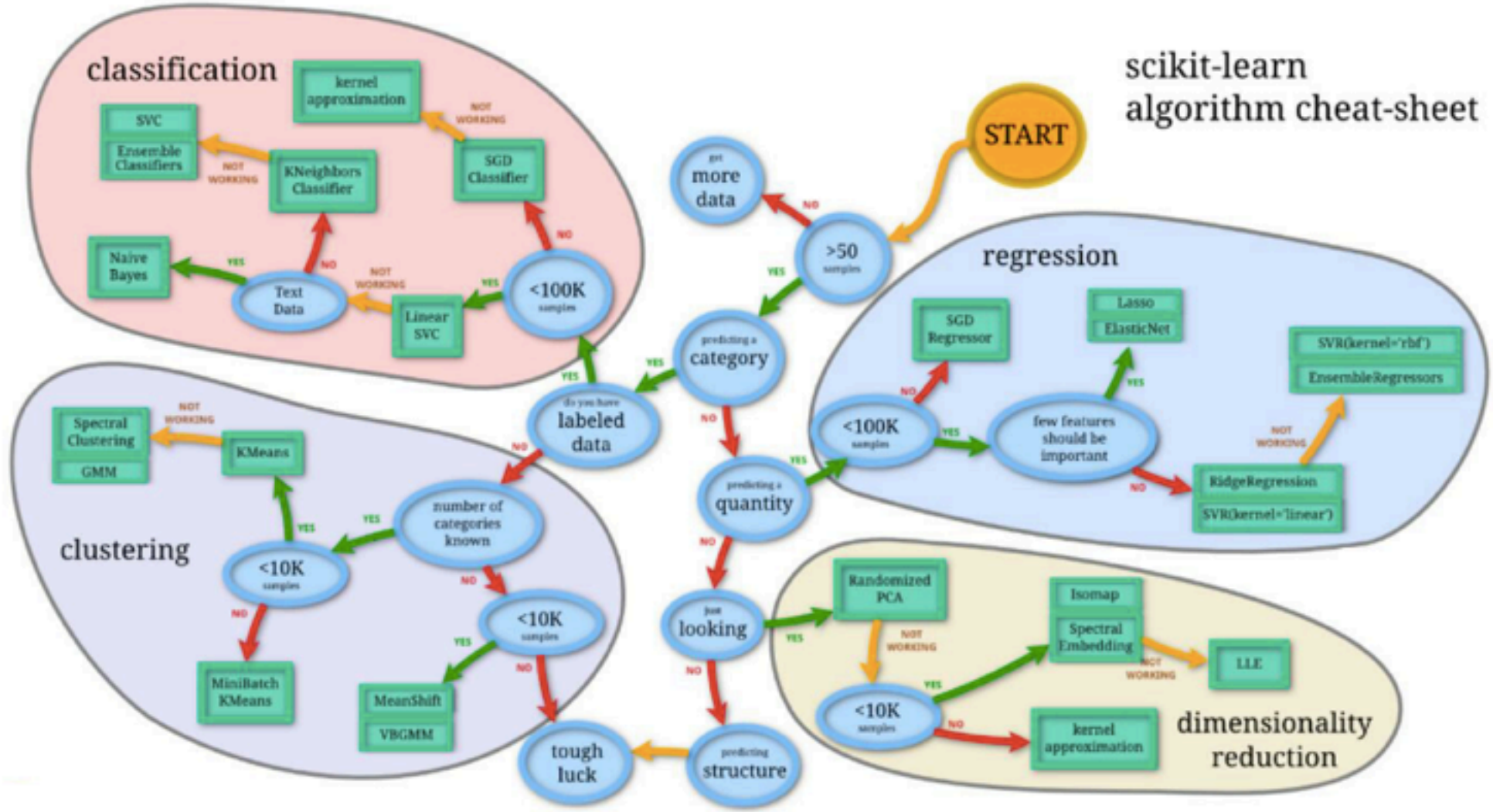
```
#1. 引入库，导入数据进行训练
from sklearn.cluster import DBSCAN
dbs = DBSCAN(eps = 1.4, min_samples = 6)
dbs.fit(d)
```

```
#2. 处理数据
a = d[dbs.labels_==-1]
a_x = a[:,0]
a_y = a[:,1]
b = d[dbs.labels_==0]
b_x = b[:,0]
b_y = b[:,1]
c = d[dbs.labels_==1]
c_x = c[:,0]
c_y = c[:,1]
e = d[dbs.labels_==2]
e_x = e[:,0]
e_y = e[:,1]
```

```
#3. 绘制图形
plt.scatter(a_x,a_y,c=‘red’)
plt.scatter(b_x,b_y,c=‘green’)
plt.scatter(c_x,c_y,c=‘blue’)
plt.scatter(e_x,e_y,c=‘pink’)
plt.show()
```

目录

模型选择评估


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
SVC
Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K
more data
>50
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
<100K
few features should be important
RidgeRegression
SVR(kernel='linear')
predicting a category
do you have labeled data
predicting a quantity
clustering
Spectral Clustering
GMM
KMeans
number of categories known
<10K
MiniBatch KMeans
<10K
MeanShift
VBGMM
just looking
Randomized PCA
Isomap
Spectral Embedding
LLE
<10K
kernel approximation
dimensionality reduction
tough luck
predicting structure
YES
NO
NOT WORKING

Thank you !