KNN算法

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

学习目标

Learning Objectives

1. 理解K近邻算法的思想
2. 知道K近邻算法分类流程
3. 知道K近邻算法回归流程

KNN

- K-近邻算法（K Nearest Neighbor，简称KNN）。比如：根据你的“邻居”来推断出你的类别

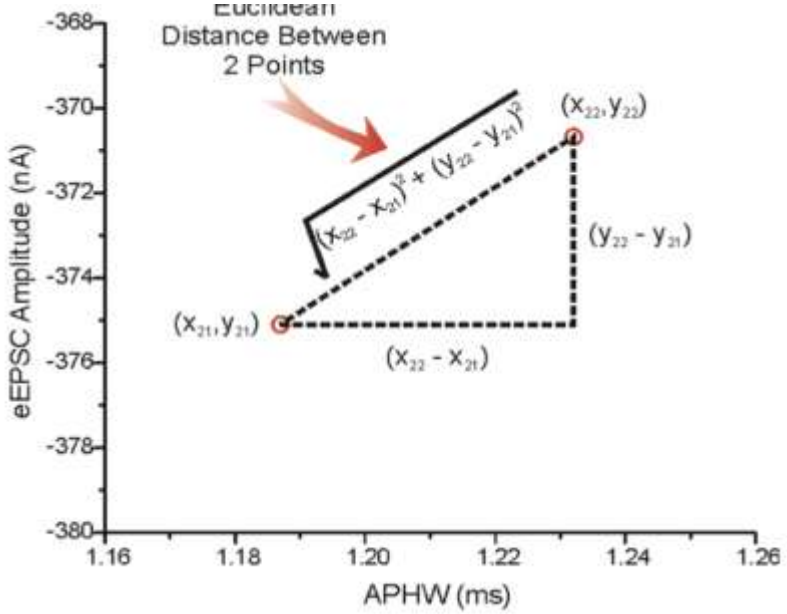
- KNN算法思想：如果一个样本在特征空间中的 k 个最相似的样本中的大多数属于某一个类别，则该样本也属于这个类别

思考：如何确定样本的相似性？

K-近邻算法

- 样本相似性：样本都是属于一个任务数据集的。样本距离越近则越相似。
- 利用K近邻算法预测电影类型

序号	电影名称	搞笑镜头	拥抱镜头	打斗镜头	电影类型
1	功夫熊猫	39	0	31	喜剧片
2	叶问3	3	2	65	动作片
3	伦敦陷落	2	3	55	动作片
4	代理情人	9	38	2	爱情片
5	新步步惊心	8	34	17	爱情片
6	谍影重重	5	2	57	动作片
7	功夫熊猫	39	0	31	喜剧片
8	美人鱼	21	17	5	喜剧片
9	宝贝当家	45	2	9	喜剧片
10	唐人街探案	23	3	17	?



二维平面上点$a(x_1,y_1)$与$b(x_2,y_2)$间的欧氏距离:

$$d_{12}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

三维空间点$a(x_1,y_1,z_1)$与$b(x_2,y_2,z_2)$间的欧氏距离:

$$d_{12}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$$

n维空间点$a(x_{11},x_{12},\ldots,x_{1n})$与$b(x_{21},x_{22},\ldots,x_{2n})$间的欧氏距离（两个n维向量）：

$$d_{12}=\sqrt{\sum_{k=1}^{n}(x_{1k}-x_{2k})^2}$$

差方和开根号

K-近邻算法

算距离，找前几个邻居

序号	电影名称	搞笑镜头	拥抱镜头	打斗镜头	电影类型	距离	K=5时
1	功夫熊猫	39	0	31	喜剧片	21.47	√
2	叶问3	3	2	65	动作片	52.01	
3	伦敦陷落	2	3	55	动作片	43.42	
4	代理情人	9	38	2	爱情片	40.57	
5	新步步惊心	8	34	17	爱情片	34.44	√
6	谍影重重	5	2	57	动作片	43.87	
7	功夫熊猫	39	0	31	喜剧片	21.47	√
8	美人鱼	21	17	5	喜剧片	18.55	√
9	宝贝当家	45	2	9	喜剧片	23.43	√
10	唐人街探案	23	3	17	?	——	?

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

《唐人街探案》
VS
《伦敦陷落》

$$d = \sqrt{(23-2)^2 + (3-3)^2 + (17-55)^2} = 43.42$$

分别计算10号电影与前9个电影的距离

k=5找到最相似的电影类别，来决定10号电影的类别

K值选择

序号	电影名称	搞笑镜头	拥抱镜头	打斗镜头	电影类型	距离	K=5时
1	功夫熊猫	39	0	31	喜剧片	21.47	√
2	叶问3	3	2	65	动作片	52.01	
3	二次曝光	2	3	55	爱情片	43.42	
4	代理情人	9	38	2	爱情片	40.57	√
5	新步步惊心	8	34	17	爱情片	34.44	√
6	谍影重重	5	2	57	动作片	43.87	
7	美人鱼	21	17	5	喜剧片	18.55	√
8	宝贝当家	45	2	9	喜剧片	23.43	√
9	唐人街探案	23	3	17	?	——	?

- K值过小：用较小邻域中的训练实例进行预测

  容易受到异常点的影响

  K值的减小就意味着整体模型变得复杂，容易发生过拟合

- 举例：K=N（N为训练样本个数）

  无论输入实例是什么，只会按训练集中最多的类别进行预测，

  受到样本均衡的影响

- K值过大：用较大邻域中的训练实例进行预测

  受到样本均衡的问题

  且K值的增大就意味着整体的模型变得简单，欠拟合

- 如何对K超参数进行调优？

  需要一些方法来寻找这个最合适的K值

  交叉验证、网格搜索

KNN算法

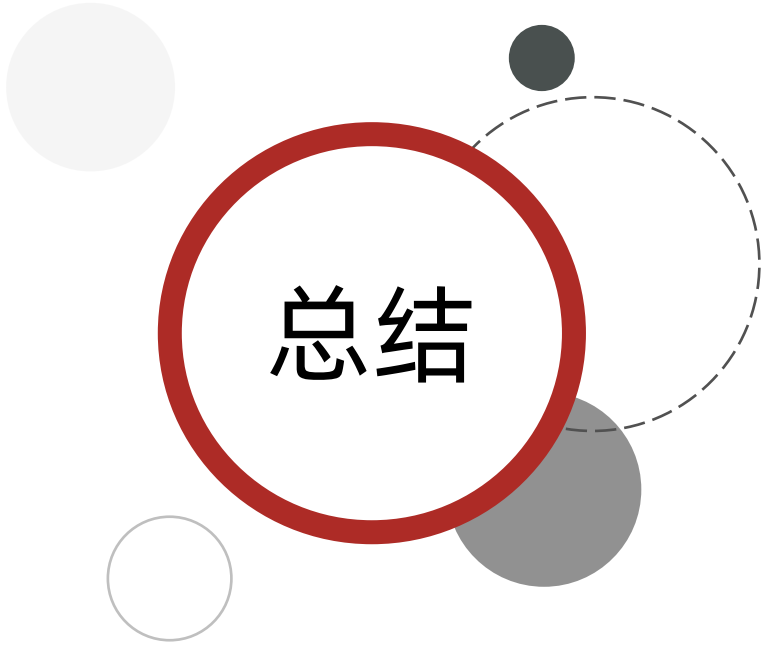
- 解决问题：分类问题、回归问题
- 算法思想：若一个样本在特征空间中的 k 个最相似的样本大多数属于某一个类别，则该样本也属于这个类别
- 相似性：欧氏距离

分类流程

1.计算未知样本到每一个训练样本的距离
2.将训练样本根据距离大小升序排列
3.取出距离最近的 K 个训练样本
4.进行多数表决，统计 K 个样本中哪个类别的样本个数最多
5.将未知的样本归属到出现次数最多的类别

回归流程

1.计算未知样本到每一个训练样本的距离
2.将训练样本根据距离大小升序排列
3.取出距离最近的 K 个训练样本
4.把这个 K 个样本的目标值计算其平均值
5.将上述平均值作为将未知的样本预测的值



1 KNN概念（K Nearest Neighbor）

- 一个样本最相似的 k 个样本中的大多数属于某一个类别，则该样本也属于这个类别

2 KNN分类流程

1.计算未知样本到每一个训练样本的距离
2.将训练样本根据距离大小升序排列
3.取出距离最近的 K 个训练样本
4.进行多数表决，统计 K 个样本中哪个类别的样本个数最多
5.将未知的样本归属到出现次数最多的类别

3 KNN回归流程

1.计算未知样本到每一个训练样本的距离
2.将训练样本根据距离大小升序排列
3.取出距离最近的 K 个训练样本
4.把这个 K 个样本的目标值计算其平均值
5.将未知的样本预测的值了

4 K值的选择

- K值过小：过拟合
  意味着模型更易受到异常点影响，更易学习到嘈杂数据，模型有过拟合的风险
- K值过大：欠拟合
  意味着模型变的复杂了，不易受异常点影响，模型有欠拟合风险

练习

1、有关KNN的K值选择，以下说法中正确的是？（**多选**）

A）若k值过小，意味着模型更易受到异常点影响，更易学习到嘈杂数据，模型有过拟合的风险。

B）若k值过大，模型会变的相对简单，结果更容易受到异常值的影响。

C）若k值与训练集样本数相同，会导致最终模型的结果都是指向训练集中类别数最多的那一类，忽略了数据当中其它的重要信息，模型会过于简单。

D）实际工作中经常使用交叉验证的方式去选取最优的k值，而且一般情况下，k值都是比较小的数值。

答案 ACD

练习

- **KNN解决什么问题**

1）KNN：K Nearest Neighbor。

即解决的是寻找与未知样本 _____ 的K个样本，并对未知样本所属的分类或者属性进行预测的问题。

2）距离度量：

空间中两个样本的距离通过 _____ 来度量的。

答案解析：① 最近邻；② 欧氏距离

目录

Contents

学习目标

Learning Objectives

1. 掌握KNN算法分类API
2. 掌握KNN算法回归API

练习

KNN算法API使用 - 分类问题

- KNN分类API

  sklearn.neighbors.KNeighborsClassifier(n_neighbors=5)

  n_neighbors：int,可选（默认= 5），k_neighbors查询默认使用的邻居数

```
from sklearn.neighbors import KNeighborsClassifier
def dm01_knnapi_分类():
    estimator =  KNeighborsClassifier(n_neighbors=1)
    X = [[0], [1], [2], [3]]
    y = [0, 0, 1, 1]
    estimator.fit(X, y)
    myret = estimator.predict([[4]])
    print('myret-->', myret)
```

练习

KNN算法API使用 - 回归问题

KNN回归API：

`sklearn.neighbors.KNeighborsRegressor(n_neighbors=5)`

```
from sklearn.neighbors import KNeighborsRegressor
def dm02_knnapi_回归():
    estimator =  KNeighborsRegressor(n_neighbors=2)
    X = [[0, 0, 1],
         [1, 1, 0],
         [3, 10, 10],
         [4, 11, 12]]
    y = [0.1, 0.2, 0.3, 0.4]
    estimator.fit(X, y)
    myret = estimator.predict([[3, 11, 10]])
    print('myret-->', myret)
```

练习

以下关于KNN算法API的使用有误的一项是？

1）导入算法对象：

```
from sklearn.neighbors import KNeighborsClassifier
```
---------- **A**

2）构造数据：

```
x = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
```

3）实例化算法API：

```
estimator = KNeighborsClassifier(n_neighbors=2)
```
---------- **B**

4）训练模型并预测：

```
estimator.fit(x, y)
```
---------- **C**

```
estimator.predict([1])
```
---------- **D**

答案解析：D　　正确代码应为：estimator.predict([[1]])

目录

Contents

学习目标

Learning Objectives

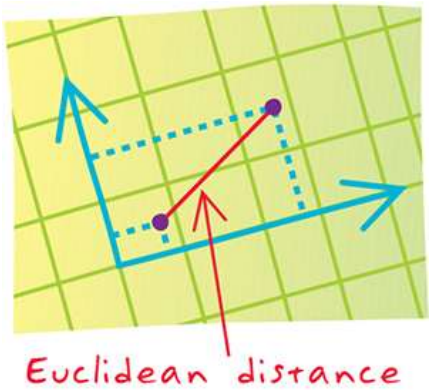
1. 掌握欧氏距离的计算方法
2. 掌握曼哈顿距离的计算方法
3. 了解切比雪夫距离的计算方法
4. 了解闵可夫斯基距离的计算方法

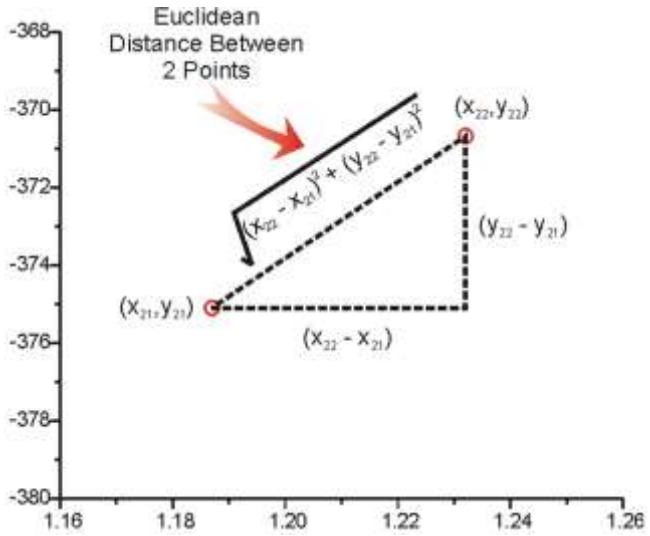
距离度量 distance measure - 常见距离公式

- 欧氏距离 Euclidean Distance

  直观的距离度量方法，

  两个点在空间中的距离一般都是指欧氏距离





二维平面上点$a(x_1,y_1)$与$b(x_2,y_2)$间的欧氏距离:

$$d_{12}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

三维空间点$a(x_1,y_1,z_1)$与$b(x_2,y_2,z_2)$间的欧氏距离:

$$d_{12}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$$

n维空间点$a(x_{11},x_{12},\ldots,x_{1n})$与$b(x_{21},x_{22},\ldots,x_{2n})$间的欧氏距离（两个n维向量）：

$$d_{12}=\sqrt{\sum_{k=1}^{n}(x_{1k}-x_{2k})^2}$$

举个例子：ABCD四点 X=[[1,1],[2,2],[3,3],[4,4]]， 计算AB AC AD BC BD CD距离

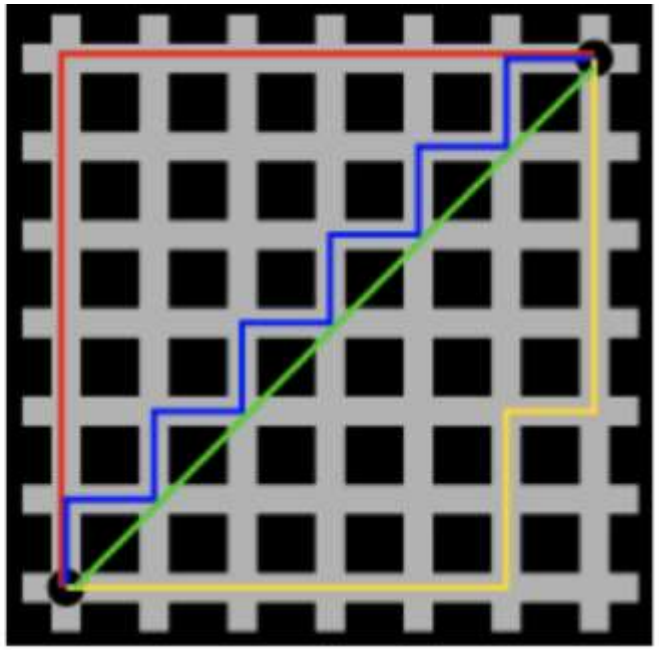
经计算得: AB = $\sqrt{(2-1)^2+(2-1)^2}=\sqrt{2}$ = 1.414

d = 1.4142 2.8284 4.2426 1.4142 2.8284 1.4142

距离度量 distance measure – 常见距离公式

- 曼哈顿距离(Manhattan Distance)

  也称为“城市街区距离”(City Block distance)，曼哈顿城市特点：横平竖直



举个例子：

ABCD四点 X=[[1,1], [2,2], [3,3], [4,4]] ，

计算AB AC AD BC BD CD曼哈顿距离

经计算得: AB =|2 − 1| + [2 − 1]= 2

d = 2 4 6 2 4 2

二维平面两点a(x_1,y_1)与b(x_2,y_2)间的曼哈顿距离 $d_{12} = |x_1 - x_2| + |y_1 - y_2|$ 绝对差之和

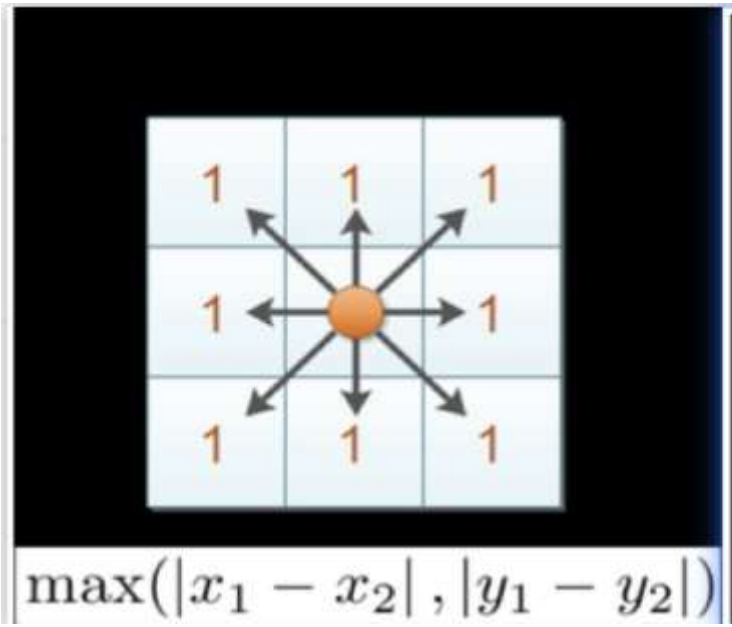
n维空间点a(x_{11},x_{12},...,x_{1n})与b(x_{21},x_{22},...,x_{2n})的曼哈顿距离 $d_{12} = \sum_{k=1}^{n} |x_{1k} - x_{2k}|$

距离度量 distance measure - 常见距离公式

- 切比雪夫距离 Chebyshev Distance

  国际象棋中，国王可以直行、横行、斜行，所以国王走一步可以移动到相邻8个方格中的任意一个。

  国王从格子(x1,y1)走到格子(x2,y2)最少需要多少步？这个距离就叫切比雪夫距离。



举个例子：

ABCD四点 X=[[1,1], [2,2], [3,3], [4,4]] ，

计算AB AC AD BC BD曼哈顿距离

经计算得: AB =max(|2 − 1|, |2 − 1|) = 1

d = 1 2 3 1 2 1

二维平面两点a(x_1,y_1)与b(x_2,y_2)间的切比雪夫距离 $d_{12} = \max(|x_1 - x_2|, |y_1 - y_2|)$

n维空间点a(x_{11},x_{12},...,x_{1n})与b(x_{21},x_{22},...,x_{2n})的切比雪夫距离 $d_{12} = \max(|x_{1i} - x_{2i}|)$

绝对差 最大值

距离度量 distance measure - 常见距离公式

- 闵可夫斯基距离 Minkowski Distance 闵氏距离
  - 不是一种新的距离的度量方式。
  - 是对多个距离度量公式的概括性的表述

- 两个n维变量a(x11 ,x12, ..., x1n) 与 b(x21, x22,..., x2n)间的闵可夫斯基距离定义为：

$$d_{12} = \sqrt[p]{\sum_{k=1}^{n} |x_{1k} - x_{2k}|^p}$$

其中p是一个变参数：

当 p=1 时，就是曼哈顿距离；

当 p=2 时，就是欧氏距离；

当 p→∞ 时，就是切比雪夫距离

根据 p 的不同，闵氏距离可表示某一类种的距离

练习

1、对于n维空间中的两个点 a(x_{11}, x_{12}, ..., x_{1n}) 和 b(x_{21}, x_{22}, ..., x_{2n}) 来说，下列哪个公式是闵可夫斯基距离的一般表达式？

A: $\sum_{k=1}^{n}|x_{1k}-x_{2k}|$

C: $\max(|x_{1i}-x_{2i}|)$

B: $\sqrt{\sum_{k=1}^{n}(x_{1k}-x_{2k})^2}$

D: $d_{12}=\sqrt[p]{\sum_{k=1}^{n}|x_{1k}-x_{2k}|^p}$

其中A是曼哈顿距离，B是欧氏距离，C是切比雪夫距离。

正确答案：D

目录

Contents

学习目标

Learning Objectives

1. 知道为什么进行归一化、标准化
2. 能应用归一化API处理数据
3. 能应用标准化API处理数据
4. 使用KNN算法进行鸢尾花分类

特征预处理

- 为什么做归一化和标准化？

特征的**单位或者大小相差较大，或者某特征的方差相比其他的特征要大出几个数量级，容易影响（支配）目标结果，**使得一些模型（算法）无法学习到其它的特征。

编号	身高 m	体重 kg	视力 0.2-2.0	健康状况
1	1.70	67	1.5	1
2	1.71	80	0.8	2
3	1.75	70	1.5	1
4	1.76	68	1.2	1
5	1.80	80	1.8	1
6	1.81	90	0.6	2
男子身高体重视力与健康关系表 健康为1 不健康为2				

特征预处理

- 归一化：通过对原始数据进行变换把数据映射到【mi,mx】(默认为[0,1])之间

$$X' = \frac{x - min}{max - min} \qquad X'' = X' * (mx - mi) + mi$$

特征1	特征2	特征3	特征4
90	2	10	40
60	4	15	45
75	3	13	46

特征1	特征2	特征3	特征4
$\frac{90-60}{90-60}$	$\frac{2-2}{4-2}$	$\frac{10-10}{15-10}$	$\frac{40-40}{46-40}$
$\frac{60-60}{90-60}$	$\frac{4-2}{4-2}$	$\frac{15-10}{15-10}$	$\frac{45-40}{46-40}$
$\frac{75-60}{90-60}$	$\frac{3-2}{4-2}$	$\frac{13-10}{15-10}$	$\frac{46-40}{46-40}$

注：里面是第一步，还需要第二步乘以(1-0)+0

特征1	特征2	特征3	特征4
1.	0.	0.	0.
0.	1.	1.	0.83
0.5	0.5	0.6	1.

特征预处理

- 数据归一化：通过对原始数据进行变换把数据映射到【mi,mx】(默认为[0,1])之间

数据归一化API：

1.sklearn.preprocessing.MinMaxScaler (feature_range=(0,1)...)

feature_range 缩放区间

2. fit_transform(X) 将特征进行归一化缩放

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

def dm01_MinMaxScaler():
 # 1. 准备数据
   data = [[90, 2, 10, 40],
           [60, 4, 15, 45],
           [75, 3, 13, 46]]

   # 2. 初始化归一化对象
   transformer = MinMaxScaler()

   # 3. 对原始特征进行变换
   data = transformer.fit_transform(data)

   # 4. 打印归一化后的结果
   print(data)
```

特征预处理

- 数据标准化：通过对原始数据进行标准化，转换为均值为0，标准差为1的标准正态分布的数据

$$X' = \frac{x - \text{mean}}{\sigma}$$

mean 为特征的平均值

σ 为特征的标准差

数据标准化API：

1.sklearn.preprocessing. StandardScaler()

2. fit_transform(X) 将特征进行标准化缩放

```
from sklearn.preprocessing import StandardScaler

def dm03_StandardScaler():     # 对特征值进行标准化
    # 1. 准备数据
    data = [[90, 2, 10, 40],
            [60, 4, 15, 45],
            [75, 3, 13, 46]]

    # 2. 初始化标准化对象
    transformer = StandardScaler()

    # 3. 对原始特征进行变换
    data = transformer.fit_transform(data)

    # 4. 打印归一化后的结果
    print(data)

    # 5 打印每1列数据的均值和标准差
    print('transformer.mean_-->', transformer.mean_)
    print('transformer.var_-->', transformer.var_)
```

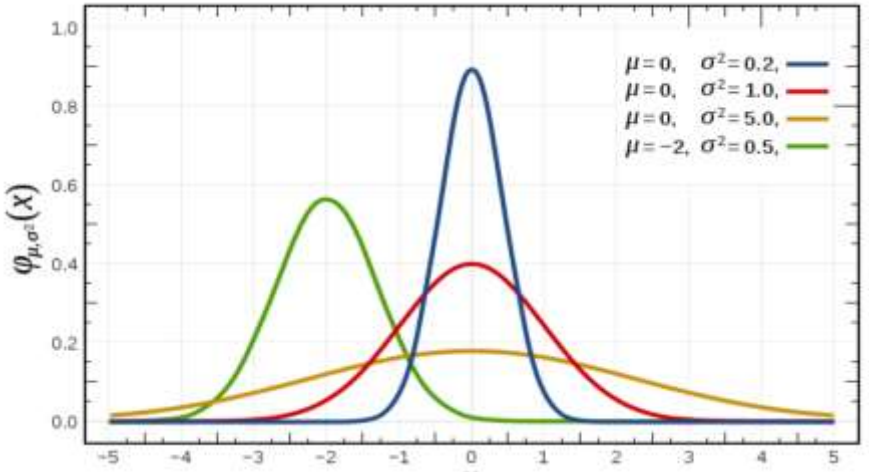
特征预处理

- 正态分布是一种概率分布，大自然很多数据符合正态分布

  也叫高斯分布，钟形分布。正态分布记作N(μ，σ)

  μ决定了其位置，其标准差σ决定了分布的幅度

  当μ = 0, σ = 1时的正态分布是标准正态分布



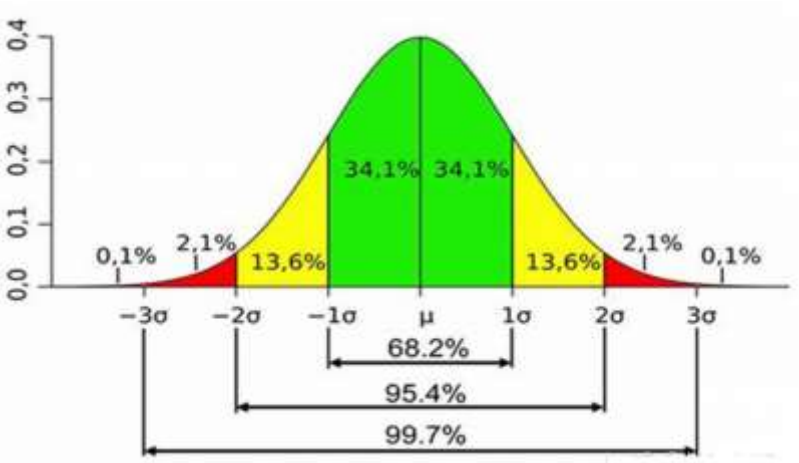
- **方差σ^2**是在概率论和统计方差衡量一组数据时离散程度的度量

  其中M为均值 n为数据总数

$$\sigma^2 = \frac{(x_1 - M)^2 + (x_2 - M)^2 + (x_3 - M)^2 + \ldots\ldots (x_n - M)^2}{n}$$

- 标准差σ是方差开根号

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$



- 正态分布的3σ法则（68-95-99.7 法则）

  68%的数据分布在（μ - σ, μ + σ)区间内；

  95%的数据分布在（μ - 2σ, μ + 2σ)区间内；

  99.7%的数据分布在（μ - 3σ, μ + 3σ)区间内。

特征预处理

数据归一化

- 如果出现异常点，影响了最大值和最小值，那么结果显然会发生改变
- 应用场景：最大值与最小值非常容易受异常点影响，鲁棒性较差，只适合传统精确小数据场景
- sklearn.preprocessing.MinMaxScaler (feature_range=(0,1)...)

数据标准化

- 如果出现异常点，由于具有一定数据量，少量的异常点对于平均值的影响并不大
- 应用场景：适合现代嘈杂大数据场景。
- sklearn.preprocessing.StandardScaler( )

练习

填空：

归一化

$$X' = \frac{x - min}{max - min}$$

$$X'' = X' * (mx - mi) + mi$$

度量值 X’ 容易受到样本中 _____ 的影响，鲁棒性差。（提示：1分布 2最大值最小值 3方差）

标准化 → $$X' = \frac{x - \text{mean}}{\sigma}$$

在 _____ 的情况下，异常值对样本的均值和标准差的影响可以忽略不计。

（提示：1样本数量小 2样本数量大）

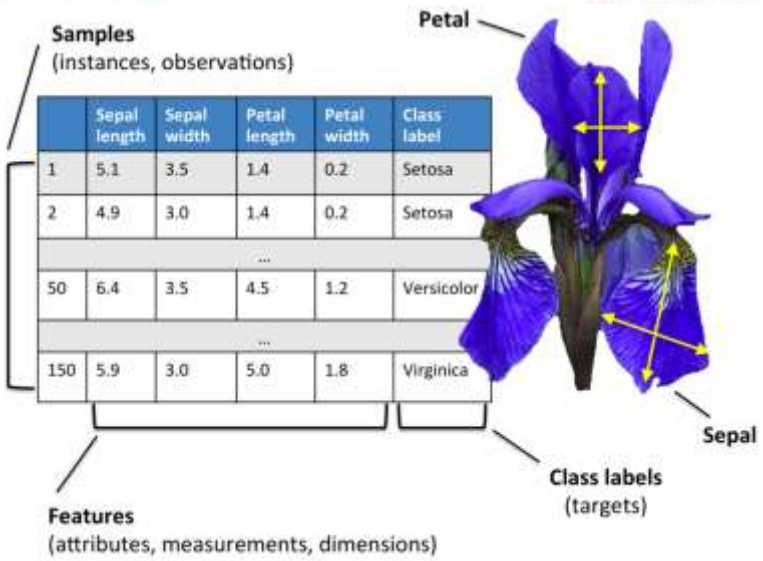
答案：① 异常值；② 样本数量较大。

案例 利用KNN算法对鸢尾花分类

变色鸢尾Iris versicolor　维吉尼亚鸢尾Iris virginica　山鸢尾 Iris Setosa

Sepal 花萼　花瓣Petal

鸢尾花 相关术语



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

实现流程：

1 获取数据集

2 数据基本处理

3 特征工程（数据标准化等）

4 模型训练

5 模型预测评估

案例

利用KNN算法对鸢尾花分类 -加载鸢尾花数据集

```
from sklearn.datasets import load_iris

# 加载鸢尾花数据集, 并显示属性
dataset.data .target .target_names .feature_names .DESCR
def dm01_loadiris():
    # 加载数据集
    mydataset = load_iris()
    # 查看数据集信息
    print('\n查看数据集信息-->\n', mydataset.data[:5])
    # 查看目标值
    print('mydataset.target-->\n', mydataset.target)
    # 查看目标值名字
    print('mydataset.target_names-->\n', mydataset.target_names)
    # 查看特征名
    print('mydataset.feature_names-->\n', mydataset.feature_names)
    # 查看数据集描述
    print('\nmydataset.DESCR-->\n', mydataset.DESCR)
    # 数据文件路径
    print('mydataset.filename-->\n', mydataset.filename)
```

案例 利用KNN算法对鸢尾花分类 -加载鸢尾花数据集

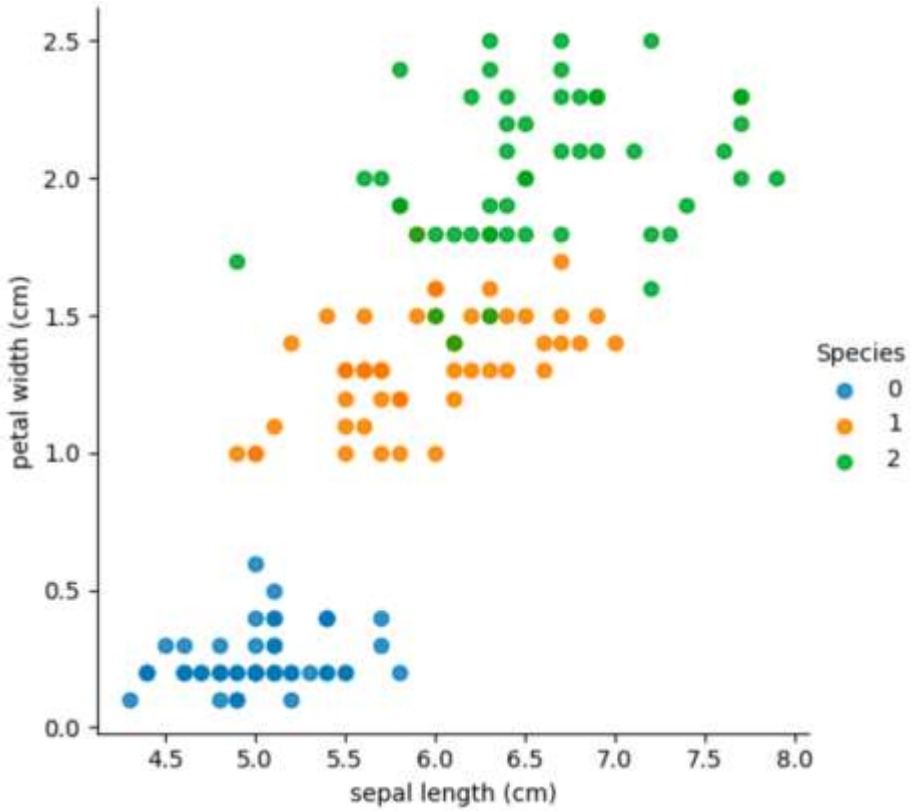
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# 显示鸢尾花数据
def dm02_showiris():

    # 1 载入鸢尾花数据集 并显示特征名称.feature_names
    mydataset = load_iris()
    print(mydataset.feature_names)

    # 2 把数据转换成dataframe格式 设置data, columns属性 目标值名称
    iris_d = pd.DataFrame(mydataset['data'], columns=mydataset.feature_names)
    iris_d['label'] = mydataset.target
    print('\niris_d-->\n', iris_d)
    col1 = 'sepal length (cm)'
    col2 = 'petal width (cm)'

    # 3 sns.lmplot()显示
    sns.lmplot(x=col1, y=col2, data=iris_d, hue='label', fit_reg=False)
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.title('iris')
    plt.show()
```



案例

利用KNN算法对鸢尾花分类 - 数据集划分

```
from sklearn.model_selection import train_test_split

# 数据集划分
def dm03_traintest_split():

    # 1 加载数据集
    mydataset = load_iris()

    # 2 划分数据集
    X_train, X_test, y_train, y_test =  train_test_split(mydataset.data, mydataset.target, test_size=0.3, random_state=22)


    print('数据总数量', len(mydataset.data))
    print('训练集中的x-特征值', len(X_train))
    print('测试集中的x-特征值', len(X_test))
    print(y_train)
```

案例

利用KNN算法对鸢尾花分类 - 模型训练

```
# 导入类库
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
def dm04_模型训练和预测():
    # 1 获取数据集
    mydataset = load_iris()
    # 2 数据基本处理
    x_train, x_test, y_train, y_test =  train_test_split(mydataset.data, mydataset.target, test_size=0.2,
random_state=22)
    # 3 数据集预处理-数据标准化
    transfer = StandardScaler()
    x_train = transfer.fit_transform(x_train)
    # 让测试集的均值和方差, 转换测试集数据;
    x_test = transfer.transform(x_test)
    # 4 机器学习(模型训练)
    estimator = KNeighborsClassifier(n_neighbors=3)
    estimator.fit(x_train, y_train)
    # 5 模型评估 直接计算准确率 100个样本中模型预测对了多少
    myscore = estimator.score(x_test, y_test)
    print('myscore-->', myscore)

    # 6 模型预测 # 需要对待预测数据,执行标准化
    print('通过模型查看分类类别-->', estimator.classes_)
    mydata =  [[5.1, 3.5, 1.4, 0.2],
               [4.6, 3.1, 1.5, 0.2]]
    mydata = transfer.transform(mydata)
    print('mydata-->', mydata)
    mypred = estimator.predict(mydata)
    print('mypred-->\n', mypred)
    mypred = estimator.predict_proba(mydata)
    print('mypred-->\n', mypred)
```

案例

利用KNN算法对鸢尾花分类 - 模型评估

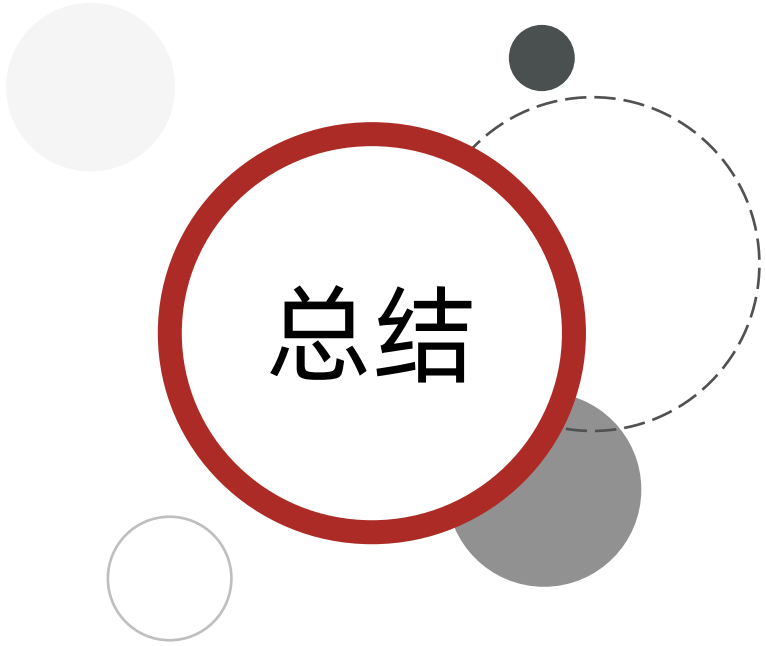
```
# 1 获取数据集
mydataset = load_iris()

# 2 数据基本处理
x_train, x_test, y_train, y_test = train_test_split(mydataset.data, mydataset.target, test_size=0.2, random_state=22)

# 3 数据集预处理-数据标准化
transfer = StandardScaler()
x_train = transfer.fit_transform(x_train)
# 让测试集的均值和方法, 转换测试集数据;
x_test = transfer.transform(x_test)

# 4 机器学习(模型训练)
estimator = KNeighborsClassifier(n_neighbors=3)
estimator.fit(x_train, y_train)

# 5-1 直接使用score函数 模型评估 100个样本中模型预测对了多少
myscore = estimator.score(x_test, y_test)
print('myscore-->', myscore)
# 5-2 利用sklearn.metrics包中的accuracy_score 方法
y_predict = estimator.predict(x_test)
myresult = accuracy_score(y_test, y_predict)
print('myresult-->', myresult)
```



1 鸢尾花数据集下载和使用

- 加载数据集mydataset = load_iris()
- 数据集属性 dataset.data .target .target_names .feature_names .DESCR

2 案例的总体处理流程

1 获取数据集
2 数据基本处理
3 特征工程
4 机器学习(模型训练)
5 模型评估

3 使用可视化加载和探索数据，以确定特征是否能将不同类别分开

4 通过标准化特征，并随机抽样到训练集和测试集来准备数据。

5 通过统计学，利用准确率评估机器学习模型

目录

Contents

学习目标

Learning Objectives

1. 知道交叉验证是什么?
2. 知道网格搜索是什么?
3. 知道交叉验证网格搜索API函数用法
4. 实践交叉验证网格搜索进行模型超参数调优
5. 利用KNN算法实现手写数字识别

交叉验证（Cross Validation）

- 什么是交叉验证？

  是一种数据集的分割方法，将训练集划分为 n 份，拿一份做验证集（测试集）、其他n-1份做训练集

- 交叉验证法原理：将数据集划分为 cv=4 份

  1. 第一次：把第一份数据做验证集，其他数据做训练
  2. 第二次：把第二份数据做验证集，其他数据做训练
  3. ... 以此类推，总共训练4次，评估4次。
  4. 使用训练集+验证集多次评估模型，取平均值做交叉验证为模型得分
  5. 若k=5模型得分最好，再使用全部数据集(训练集+验证集) 对k=5模型再训练一边，再使用测试集对k=5模型做评估

交叉验证法，是划分数据集的一种方法，目的就是为了得到更加准确可信的模型评分。

网格搜索

- 为什么需要网格搜索？
  - 模型有很多超参数，其能力也存在很大的差异。需要手动产生很多超参数组合，来训练模型
  - 每组超参数都采用交叉验证评估，最后选出最优参数组合建立模型。

- 网格搜索是模型调参的有力工具。寻找最优超参数的工具！

  只需要将若干参数传递给网格搜索对象，它自动帮我们完成不同超参数的组合、模型训练、模型评估，最终返回一组最优的超参数。

- **网格搜索 + 交叉验证的强力组合 (模型选择和调优)**
  - 交叉验证解决模型的数据输入问题(数据集划分)得到更可靠的模型
  - 网格搜索解决超参数的组合
  - 两个组合再一起形成一个模型参数调优的解决方案

交叉验证网格搜索 - API和应用举例

- 交叉验证网格搜索API介绍

  - sklearn.model_selection.GridSearchCV(estimator, param_grid=None,cv=None)
    - 对估计器的指定参数值进行详尽搜索
    - estimator：估计器对象
    - param_grid：估计器参数(dict){“n_neighbors”:[1,3,5]}
    - cv：指定几折交叉验证
    - fit：输入训练数据
    - score：准确率
    - 结果分析：
      - best_score_ ：在交叉验证中验证的最好结果
      - best_estimator_：最好的参数模型
      - cv_results_ :每次交叉验证后的验证集准确率结果和训练集准确率结果

*输入一个**estimator** 返回一个**estimator***
*此**estimator**会更强大*
拥有交叉验证网格搜索的功能

案例 利用KNN算法对鸢尾花分类 - 交叉验证网格搜索

```
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier

def dm01_鸢尾花knn分类_交叉验证网格搜索():
    # 1 获取数据集
    mydataset = load_iris()

    # 2 数据基本处理-划分数据集
    x_train, x_test, y_train ,y_test = train_test_split(mydataset.data, mydataset.target,
test_size=0.2,random_state=22)

    # 3 数据集预处理-数据标准化
    transfer = StandardScaler()
    x_train = transfer.fit_transform(x_train)
    x_test = transfer.transform(x_test)

    # 4 机器学习(模型训练)
    estimator = KNeighborsClassifier()
    print('estimator-->', estimator)
```

```
    # 4-2 使用校验验证网格搜索
    param_grid = {'n_neighbors':[1,3,5,7]}
    # 输入一个estimator, 出来一个estimator(功能变的强大)
    estimator = GridSearchCV(estimator=estimator, param_grid=param_grid, cv=5)
    estimator.fit(x_train, y_train) # 4个模型 每个模型进行网格搜索找到做好的模型

    # 4-3 交叉验证网格搜索结果查看
    # estimator.best_score_ .best_estimator_ .best_params_ .cv_results_
    print('estimator.best_score_---', estimator.best_score_)
    print('estimator.best_estimator_---', estimator.best_estimator_)
    print('estimator.best_params_---', estimator.best_params_)
    print('estimator.cv_results_---', estimator.cv_results_)

    # 4-4 保存交叉验证结果
    myret = pd.DataFrame(estimator.cv_results_)
    myret.to_csv(path_or_buf='./mygridsearchcv.csv')

    # 5 模型评估
    myscore = estimator.score(x_test, y_test)
    print('myscore-->', myscore)
```

练习

1、交叉验证和网格搜索的目的是什么？（多选题）

A）为了让被评估的模型更加准确可信，一般会使用交叉验证网格搜索去完成任务。

B）有些算法模型本身自带较多的超参数，无法高效的去筛选比较合适的超参数组合。

C）使用交叉验证和网格搜索可以提升模型的可信度和查找最佳参数组合的效率。

D）仅交叉验证功能能够提升模型的准确率

正确答案：ABC

案例 利用KNN算法实现手写数字识别

- 已知数据
  - MNIST手写数字识别
  - 1999年发布，成为分类算法基准测试的基础
  - MNIST仍然是研究人员和学习者的可靠资源
- 需求
  - 从数万个手写图像的数据集中正确识别数字

案例 利用KNN算法实现手写数字识别

- 数据介绍

1 数据文件 train.csv 和 test.csv 包含从 0 到 9 的手绘数字的灰度图像。

2 每个图像高 28 像素，宽28 像素，共784个像素。

3 每个像素取值范围[0,255]，取值越大意味着该像素颜色越深

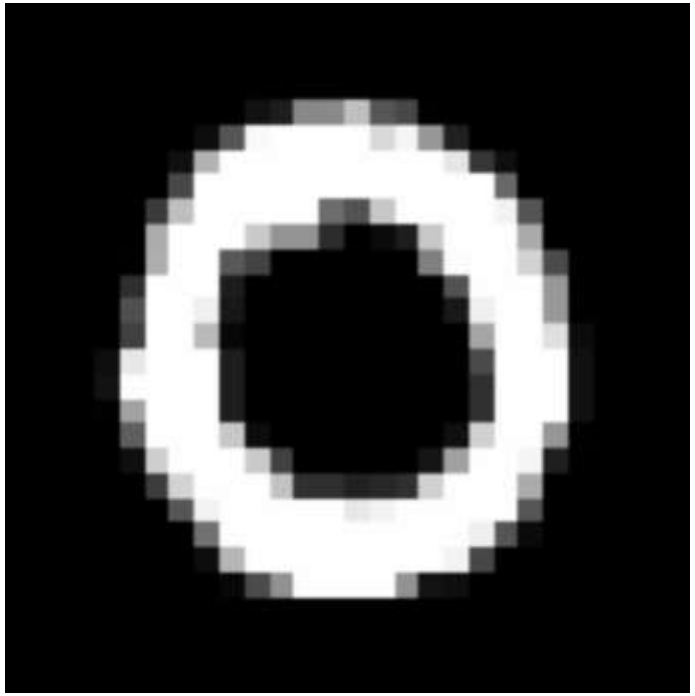
4 训练数据集（train.csv）共785列。

第一列为 "标签"，为该图片对应的手写数字。其余784列为该图像的像素值

5 训练集中的特征名称均有pixel前缀，后面的数字（[0,783])代表了像素的序号。

A	B	C	D	E	F	G	H	I	FQ	FR	FS	FT	FU	FV	FW	FX
label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel171	pixel172	pixel173	pixel174	pixel175	pixel176	pixel177	pixel178
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	16	179	254	254
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	28	247	17	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	11	208	253	253
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	130
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	135	225	244	253	202
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	64	191	255	255	255	191	0	0
3	0	0	0	0	0	0	0	0	0	0	0	13	156	252	253	233
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	23	211	253	253

案例 利用KNN算法实现手写数字识别



```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import joblib
from collections import Counter

def show_digit(idx):
    # 1 加载数据
    data = pd.read_csv('data/手写数字识别.csv')

    # 2 打印数据基本信息
    x = data.iloc[:, 1:]
    y = data.iloc[:, 0]
    print('数据基本信息:', x.shape)
    print('类别数据比例:', Counter(y))

    print('当前数字的标签为:',y[idx])

    # 3 显示指定的图片 # data修改为ndarray 类型
    data_ = x.iloc[idx].values
    # 将数据形状修改为 28*28
    data_ = data_.reshape(28, 28)
    plt.imshow(data_,  cmap='gray')
    plt.show()
```

案例 利用KNN算法实现手写数字识别

- 训练模型，程序效果

```
def train_model():
    # 1 加载手写数字数据集
    data = pd.read_csv('data/手写数字识别.csv')

    # 2 数据预处理 归一化
    x = data.iloc[:, 1:] / 255
    y = data.iloc[:, 0]

    # 3 分割数据集
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=0)

    # 4 模型训练
    estimator = KNeighborsClassifier(n_neighbors=3)
    estimator.fit(x_train, y_train)

    # 5 模型评估
    acc = estimator.score(x_test, y_test)
    print('测试集准确率: %.2f' % acc)

    # 6 模型保存
    joblib.dump(estimator, 'knn.pth')
```

案例 利用KNN算法实现手写数字识别

- 模型测试，程序效果

```
def test_model():
    # 1 读取图片数据
    img = plt.imread('data/demo.png')
    plt.imshow(img)
    plt.show()

    # 2 加载模型
    knn = joblib.load('knn.pth')

    # 3 预测图片
    y_pred = knn.predict(img.reshape(1, -1))
    print('您绘制的数字是:', y_pred)
```


黑马程序员
www.itheima.com
传智教育旗下高端IT教育品牌