

KNN,k-nearest neighbor,K近邻算法。当需要表示一个样本（值）的时候，就使用与该样本最近的K个邻居来决定（K的取值会影响结果）。KNN既可用于分类，也可用于回归。

```
In [21]: import numpy as np
import pandas as pd
```

```
In [22]: # 读取鸢尾花iris数据集，header指定标题行，默认为0
data = pd.read_csv(r"iris.csv",header = 0)
# data.head() # 显示前5行
# data.tail() # 显示末尾5行
# data.sample()# 随机抽取某行
data.sample(10)# 随机抽取10行
```

```
Out[22]:
```

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
104	105	6.5	3.0	5.8	2.2	virginica
36	37	5.5	3.5	1.3	0.2	setosa
30	31	4.8	3.1	1.6	0.2	setosa
91	92	6.1	3.0	4.6	1.4	versicolor
29	30	4.7	3.2	1.6	0.2	setosa
7	8	5.0	3.4	1.5	0.2	setosa
84	85	5.4	3.0	4.5	1.5	versicolor
112	113	6.8	3.0	5.5	2.1	virginica
144	145	6.7	3.3	5.7	2.5	virginica
109	110	7.2	3.6	6.1	2.5	virginica

```
In [23]: # 字典，map映射，将文本变为数字
data["Species"] = data["Species"].map({"versicolor":0,"setosa":1,"virginica":2})
data
```

Out[23]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	1
1	2	4.9	3.0	1.4	0.2	1
2	3	4.7	3.2	1.3	0.2	1
3	4	4.6	3.1	1.5	0.2	1
4	5	5.0	3.6	1.4	0.2	1
...
145	146	6.7	3.0	5.2	2.3	2
146	147	6.3	2.5	5.0	1.9	2
147	148	6.5	3.0	5.2	2.0	2
148	149	6.2	3.4	5.4	2.3	2
149	150	5.9	3.0	5.1	1.8	2

150 rows × 6 columns

```
In [24]: # 删除不需要的ID列,inplace=True即在原本的文档上直接修改axis默认为0,删除行;此处删除
# data.drop("Unnamed: 0",axis=1)会直接生成一个新的表data1,所以也可以data = data.c
data.drop("Unnamed: 0", axis=1, inplace=True)
data.duplicated().any()# 检查重复值;.any()返回True,表明有重复值
data.drop_duplicates(inplace=True)# 删除重复值
len(data)# 表格有多少条记录
data["Species"].value_counts()# 查看各个类别的鸢尾花具有多少条记录
```

```
Out[24]: Species
1      50
0      50
2      49
Name: count, dtype: int64
```

```
In [25]: class KNN:
    """使用python语言实现K近邻算法（实现分类）"""
    def __init__(self, k):
        """初始化方法
        Parameters
        -----
        k:int
            邻居的个数。
        """
        self.k = k

    def fit(self, X, y):# 矩阵，大写；向量，小写
        """训练方法
        Parameters
        -----
        X: 类数组类型(类似数组类型，如list,data.frame);形状为：{样本的数量149，特
            待训练的样本特征（属性）
        y: 类数组类型，形状为：{样本的数量149}
            每个样本的目标值（标签）
        """
        self.X = np.asarray(X)# 将X转换为ndarray数组形式,如果X本身就是ndarray,则
```

```

self.y = np.asarray(y)

def predict(self,X):
    """
    根据参数传递的样本，对样本数据进行预测
    Parameters
    -----
    X: 类数组类型，形状为：{样本数量，特征数量}
        待训练的样本特征（属性）
    Returns
    -----
    result: 数组类型
        预测的结果
    """

    X = np.asarray(X)
    result = []
    for x in X:# 对ndarray数组进行遍历，每次取数组中的一行（对于测试集中的每一
        dis = np.sqrt(np.sum((x - self.X)**2,axis = 1))# 按行的方式求和，再开
        index = dis.argsort()# 返回数组排序后，每个元素在排序之前的数组中的索
        # 进行截断，只取前k个元素（取距离最近的k个元素的索引）
        index = index[:self.k]# 到self.k为止
        # 返回数组中每个元素出现的次数，元素必须是非负的整数
        count = np.bincount(self.y[index])
        # 返回ndarray数组中，值最大的元素对应的索引(即出现次数最多的元素)，该
        result.append(count.argmax())
    return np.asarray(result)

```

```

In [26]: # 拆分数据集
# 提取出每个类别的鸢尾花数据
t0 = data[data["Species"] == 0]
t1 = data[data["Species"] == 1]
t2 = data[data["Species"] == 2]
# 打乱顺序,对每个类别数据进行洗牌
t0 = t0.sample(len(t0),random_state = 0)# random_state=类似于随机种子
t1 = t1.sample(len(t1),random_state = 0)
t2 = t2.sample(len(t2),random_state = 0)
# 构建训练集与测试集
train_X = pd.concat([t0.iloc[:40,:-1],t1.iloc[:40,:-1],t2.iloc[:40,:-1]],axis =
train_y = pd.concat([t0.iloc[:40,-1],t1.iloc[:40,-1],t2.iloc[:40,-1]],axis = 0)
test_X = pd.concat([t0.iloc[40:,:-1],t1.iloc[40:,:-1],t2.iloc[40:,:-1]],axis = 0
test_y = pd.concat([t0.iloc[40:,-1],t1.iloc[40:,-1],t2.iloc[40:,-1]],axis = 0)
# 创建KNN对象，进行训练与测试
knn = KNN(k = 3)
# 进行训练
knn.fit(train_X, train_y)
# 进行测试，获得测试的结果
result = knn.predict(test_X)
# display(result)
# display(test_y)
# display(result == test_y)# 检测正确情况
display(np.sum(result == test_y))# True是1, False是0，可以用np.sum直接求和
display(len(result))# 总的检测数据
display(np.sum(result == test_y)/len(result))# 预测正确比例

```

28

29

0.9655172413793104

```
In [27]: # KNN可视化
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
In [28]: # 默认情况下，matplotlib不支持中文显示，我们需要进行设置
mpl.rcParams["font.family"] = "SimHei"
# 在中文字体时，能够正常显示负号（-）
mpl.rcParams["axes.unicode_minus"] = False
```

```
In [29]: # {"versicolor":0,"setosa":1,"virginica":2}
# 设置画布大小
plt.figure(figsize=(10,10)) # 单位，英寸
# 绘制训练集数据
# Unnamed: 0    Sepal.Length    Sepal.Width    Petal.Length    Petal.Width
plt.scatter(x=t0["Sepal.Length"][:40], y=t0["Petal.Length"][:40], color="r", label="setosa")
plt.scatter(x=t1["Sepal.Length"][:40], y=t1["Petal.Length"][:40], color="g", label="versicolor")
plt.scatter(x=t2["Sepal.Length"][:40], y=t2["Petal.Length"][:40], color="b", label="virginica")
# 绘制测试集数据
right = test_X[result == test_y]
wrong = test_X[result != test_y]
plt.scatter(x=right["Sepal.Length"], y=right["Petal.Length"], color="c", marker="x", label="right")
plt.scatter(x=wrong["Sepal.Length"], y=wrong["Petal.Length"], color="m", marker=">", label="wrong")
plt.xlabel("花萼长度")
plt.ylabel("花瓣长度")
plt.title("KNN分类结果显示")
plt.legend(loc="best")
plt.show()
```

KNN分类结果显示

