# k-近邻算法也就是KNN算法

## 学习目标

|  |
| --- |
|  |

# 1.什么是KNN算法

## 1)算法图示:根据你的邻居来推断出你的类别

|  |
| --- |
|  |

## 上图中,如果蓝色那个小男孩想知道他现在身处那个区,最好的办法就是找出和他最近的那个人属于哪个区,

|  |
| --- |
|  |

## 2)KNN算法原理文字描述,k是指特征的个数

|  |
| --- |
|  |

### k其实不能取得太小,如k=1会有一些问题,如果这个k离异常值很近,可能会把特征分错类别.k也不能取得太大如k等于样本数,这样子也不合适,可能会无法确定类别或者因为样本不均衡而出现分类错误,可见KNN算法其实有很多问题.那么k应该取多大才合适呢?

### 处理欧式距离,还有另外一个叫做曼哈顿距离的算法,就是用差的绝对值来代替差的平方

## 还有一个叫做明可夫斯基距离,是欧式距离和做曼哈顿距离的推广

# 2.使用举例:电影类型分析

|  |
| --- |
|  |
|  |
|  |
|  |

# 3.算法API模块

|  |
| --- |
|  |

# 4.KNN案例1,鸢尾花种类预测

|  |
| --- |
|  |
|  |

## 实验流程

|  |
| --- |
|  |

## 实例代码如下,注意:使用sklearn自带的数据集进行学习

|  |
| --- |
|  |
| *""" KNN案例1,鸢尾花种类预测 """* from sklearn.datasets import load\_iris from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.neighbors import KNeighborsClassifier  # 1.获取数据,这里使用sklearn里面自带的数据集 iris\_data = load\_iris() # 2.划分数据集,random\_state参数值会影响KNN算法的效果 x\_train,x\_test,y\_train,y\_test = train\_test\_split(iris\_data.data, iris\_data.target,random\_state=6) # 3.特征工程:标准化处理,需要给训练集和测试集都做标准化处理 transformer = StandardScaler() x\_train = transformer.fit\_transform(x\_train) # 给训练集做标准化处理 x\_test = transformer.transform(x\_test) # 这里使用transform是因为我们需要使用训练集的平均值和标准差来对测试集做处理 # 4.KNN算法预估器 estimator = KNeighborsClassifier(n\_neighbors=3) estimator.fit(x\_train,y\_train) # 5.模型评估 # 方法1,直接比对真实值与预测值 y\_predict = estimator.predict(x\_test) print("比对真实值与预测值的结果", y\_test == y\_predict) # 方法2.计算准确率 accuracy = estimator.score(x\_test, y\_test) print("准确率为:", accuracy) |

### 效果

|  |
| --- |
|  |

# 5.KNN算法的效果以及总结

|  |
| --- |
|  |
|  |

### 使用场景:小数据场景几千-几万样本