## 1.使用knn算法+归一化

|  |
| --- |
|  |
| *""" 对第一天的dating案例的knn扩展,使用归一化处理数据 """* import pandas as pd from sklearn.preprocessing import MinMaxScaler from sklearn.neighbors import KNeighborsClassifier from sklearn.model\_selection import train\_test\_split  # 1.加载数据 data = pd.read\_csv("./dating.txt") # 2.获取数据的前三列,因为第四列是目标值不能作为特征值 features = data.iloc[:, :3] # 3.获取目标值 targets = data["target"] # print(targets) # 4.划分数据集 x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, targets, random\_state=6) # 5.创建转换器 transformer = MinMaxScaler() # 6.归一化处理 x\_train = transformer.fit\_transform(x\_train) x\_test = transformer.transform(x\_test) # 7.创建预估器 estimator = KNeighborsClassifier(n\_neighbors=3) # 2.2 训练模型,调用fit方法 estimator.fit(x\_train, y\_train) # 8模型评估 # 1>直接比对 y\_predict = estimator.predict(x\_test) print("直接比对真实值与预测值的结果:", y\_predict == y\_test) # 2>计算准确率 score = estimator.score(x\_test, y\_test) print("正确率为:", score) |

## 效果

|  |
| --- |
|  |

### 可能是因为数据集太小.结果不太正确

## 2.knn+标准化

|  |
| --- |
|  |
| *""" 对第一天的dating案例的knn扩展,使用归一化处理数据 """* import pandas as pd from sklearn.preprocessing import StandardScaler from sklearn.neighbors import KNeighborsClassifier from sklearn.model\_selection import train\_test\_split  # 1.加载数据 data = pd.read\_csv("./dating.txt") # 2.获取数据的前三列,因为第四列是目标值不能作为特征值 features = data.iloc[:, :3] # 3.获取目标值 targets = data["target"]  # 4.划分数据集 x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, targets, random\_state=6)  # print(targets) # 4.创建转换器 transformer = StandardScaler() # 5.归一化处理 x\_train = transformer.fit\_transform(x\_train) x\_test = transformer.transform(x\_test) # print(features)  # 6.创建预估器 estimator = KNeighborsClassifier(n\_neighbors=3) # 6.2 训练模型,调用fit方法 estimator.fit(x\_train, y\_train) # 7模型评估 # 1>直接比对 y\_predict = estimator.predict(x\_test) print("直接比对真实值与预测值的结果:", y\_predict == y\_test) # 2>计算准确率 score = estimator.score(x\_test,y\_test) print("正确率为:",score) |

### 效果:

|  |
| --- |
|  |

## 3.朴素贝叶斯算法+归一化

|  |
| --- |
|  |
| *""" 对第一天的dating案例的knn扩展,使用归一化处理数据 """* import pandas as pd from sklearn.preprocessing import MinMaxScaler from sklearn.naive\_bayes import MultinomialNB from sklearn.model\_selection import train\_test\_split  # 1.加载数据 data = pd.read\_csv("./dating.txt") # 2.获取数据的前三列,因为第四列是目标值不能作为特征值 features = data.iloc[:, :3] # 3.获取目标值 targets = data["target"] # print(targets) # 4.划分数据集 x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, targets, random\_state=6) # 5.创建转换器 transformer = MinMaxScaler() # 6.归一化处理 x\_train = transformer.fit\_transform(x\_train) x\_test = transformer.transform(x\_test) # 7.创建预估器 estimator = MultinomialNB() # 2.2 训练模型,调用fit方法 estimator.fit(x\_train, y\_train) # 8模型评估 # 1>直接比对 y\_predict = estimator.predict(x\_test) print("直接比对真实值与预测值的结果:", y\_predict == y\_test) # 2>计算准确率 score = estimator.score(x\_test,y\_test) print("正确率为:",score) |

### 效果

|  |
| --- |
|  |

## 如果使用朴素贝叶斯算法+标准化代码是类似的,只需要修改转换器类,但是如果又负数会抛异常