**一、简述knn算法**的原理、实现的关键操作点（不要超过**200字**，逻辑要通，**要有小标题**，要排版清晰，请不要网络大幅摘抄）

**一、原理：** KNN（K-Nearest Neighbors）是一种基本的分类和回归方法。对于分类问题，它通过测量不同特征之间的距离，将新数据点归类到与其最近邻居中占多数的类别中。对于回归问题，KNN计算与新数据点最近邻居的平均值来进行预测。二、**关键操作点**：

1.距离度量：选择合适的距离度量方法，如欧氏距离、曼哈顿距离等。2.K值选择：确定用于分类的邻居数量K，通过交叉验证或经验选择合适的K值。3.分类决策：采用投票法确定最终类别，或者对K个最近邻居的输出进行加权平均作为回归结果。

二、利用Knn算法解决问题（预测或分类）。

**1.**对于教材上**“申请国外大学研究生”项目案例**，结合**实际情况**（可以考虑我们学校以及校外机构的实际提示）进一步对**数据完善**，再**调参优化**找到合适的**Knn算法模型**。

import numpy as np  
import pandas as pd  
# data = pd.read\_csv('Admission\_Predict1.csv')  
# print(data.head())  
# print(data.info())  
# print(data.describe())  
df = pd.read\_csv('Admission\_Predict1.csv')  
df.drop(['Serial No.'],axis=1,inplace=True)  
x = df.drop(['Chance of Admit '],axis=1)  
y = df['Chance of Admit '].values  
  
from sklearn.model\_selection import train\_test\_split  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,random\_state=0)  
print('x\_train shape:{}'.format(x\_train.shape))  
print('x\_test shape:{}'.format(x\_test.shape))  
print('y\_train shape:{}'.format(y\_train.shape))  
print('y\_test shape:{}'.format(y\_test.shape))  
  
from sklearn.neighbors import KNeighborsRegressor  
reg = KNeighborsRegressor()  
reg.fit(x\_train,y\_train)  
KNeighborsRegressor(algorithm='auto',leaf\_size=30,metric='minkowski',metric\_params=None,n\_jobs=None,n\_neighbors=5,p=2,weights='uniform')  
print('\n默认参数的验证集得分：{:.2f}'.format(reg.score(x\_test,y\_test)))  
print('默认参数的训练集得分：{:.2f}'.format(reg.score(x\_train,y\_train)))  
  
reg2 = KNeighborsRegressor(n\_neighbors=2)  
reg2.fit(x\_train,y\_train)  
print('\n模型参数n\_neighbors=2的验证集得分：{:.2f}'.format(reg2.score(x\_test,y\_test)))  
print('模型参数n\_neighbors=2的训练集得分：{:.2f}'.format(reg2.score(x\_train,y\_train)))  
  
reg10 = KNeighborsRegressor(n\_neighbors=10)  
reg10.fit(x\_train,y\_train)  
print('\n模型参数n\_neighbors=10的验证集得分：{:.2f}'.format(reg10.score(x\_test,y\_test)))  
print('模型参数n\_neighbors=10的训练集得分：{:.2f}'.format(reg10.score(x\_train,y\_train)))  
  
reg\_w = KNeighborsRegressor(weights='distance')  
reg\_w.fit(x\_train,y\_train)  
print('\n模型参数weights=distance的验证集得分：{:.2f}'.format(reg\_w.score(x\_test,y\_test)))  
print('模型参数weights=distance的训练集得分：{:.2f}'.format(reg\_w.score(x\_train,y\_train)))  
  
from sklearn.preprocessing import MinMaxScaler  
x\_2 = MinMaxScaler().fit\_transform(x)  
print('\n',x\_2)  
x\_train\_pp,x\_test\_pp,y\_train,y\_test = train\_test\_split(x\_2,y,random\_state=0)  
reg\_scaled = KNeighborsRegressor()  
reg\_scaled.fit(x\_train\_pp,y\_train)  
print('数据预处理后的模型验证集得分：{:.2f}'.format(reg\_scaled.score(x\_test\_pp,y\_test)))  
print('数据预处理后的模型训练集得分：{:.2f}'.format(reg\_scaled.score(x\_train\_pp,y\_train)))  
  
x\_new = np.array([[337,118,4,4.5,4.5,9.65,1]])  
prediction = reg.predict(x\_new)  
print('\nK最近邻算法模型预测结果：')  
print("预测小A同学的综合评分为：{}".format(prediction))  
  
prediction2 = reg10.predict(x\_new)  
print('\nK最近邻算法模型预测结果(最高的分值模型)：')  
print("预测小A同学的综合评分为：{}".format(prediction2))

**结果：**

文本

描述已自动生成

2.附：Sklearn常用数据集

|  |  |  |
| --- | --- | --- |
| 序号 | 数据集 | 学号尾号 |
| 1 | 鸢尾花数据集：load\_iris() | 0 1 2 3 |
| 3 | 乳腺癌数据集：load-breast-cancer() | 4 5 6 |
| 4 | 糖尿病数据集：load-diabetes() | 7 8 9 |

利用**Knn**对以上数据集（学号尾号不同采用不同数据集）进行分析，找出更适合本数据集的Knn（n-neighbors、weights等参数调整）。

要求：

1）针对当前数据、模型，详叙数据预处理、模型参数调整过程及效果比对；

2）代码要列出，重点代码加注释说明，特别是自己调试过程中的自我理解；

3）运行结果要截图，结果要文字说明；

4）注意排版

import numpy as np  
import pandas as pd  
from sklearn import datasets  
# data = pd.read\_csv('iris.csv')  
# print(data.head())  
# print(data.info())  
# print(data.describe())  
  
# 从sklearn自带的鸢尾花数据集中加载数据  
iris = datasets.load\_iris()  
x = iris.data  
y = iris.target  
from sklearn.model\_selection import train\_test\_split  
# 将数据集划分为训练集和测试集  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=0)  
print('x\_train shape:{}'.format(x\_train.shape))  
print('x\_test shape:{}'.format(x\_test.shape))  
print('y\_train shape:{}'.format(y\_train.shape))  
print('y\_test shape:{}'.format(y\_test.shape))  
  
from sklearn.neighbors import KNeighborsRegressor  
# 使用默认参数创建K最近邻回归模型，并在训练集和测试集上进行拟合和评分  
reg = KNeighborsRegressor()  
reg.fit(x\_train, y\_train)  
print('\n默认参数的验证集得分：{:.2f}'.format(reg.score(x\_test, y\_test)))  
print('默认参数的训练集得分：{:.2f}'.format(reg.score(x\_train, y\_train)))  
  
# 创建不同参数下的K最近邻回归模型，并评估性能  
reg2 = KNeighborsRegressor(n\_neighbors=2)  
reg2.fit(x\_train, y\_train)  
print('\n模型参数n\_neighbors=2的验证集得分：{:.2f}'.format(reg2.score(x\_test, y\_test)))  
print('模型参数n\_neighbors=2的训练集得分：{:.2f}'.format(reg2.score(x\_train, y\_train)))  
  
reg10 = KNeighborsRegressor(n\_neighbors=10)  
reg10.fit(x\_train, y\_train)  
print('\n模型参数n\_neighbors=10的验证集得分：{:.2f}'.format(reg10.score(x\_test, y\_test)))  
print('模型参数n\_neighbors=10的训练集得分：{:.2f}'.format(reg10.score(x\_train, y\_train)))  
  
reg\_w = KNeighborsRegressor(weights='distance')  
reg\_w.fit(x\_train, y\_train)  
print('\n模型参数weights=distance的验证集得分：{:.2f}'.format(reg\_w.score(x\_test, y\_test)))  
print('模型参数weights=distance的训练集得分：{:.2f}'.format(reg\_w.score(x\_train, y\_train)))  
  
from sklearn.preprocessing import MinMaxScaler  
# 使用MinMaxScaler对特征进行缩放  
x\_2 = MinMaxScaler().fit\_transform(x)  
# print('\n',x\_2)  
x\_train\_pp, x\_test\_pp, y\_train, y\_test = train\_test\_split(x\_2, y, random\_state=0)  
  
# 在经过数据预处理后的数据上训练K最近邻回归模型并评估性能  
reg\_scaled = KNeighborsRegressor()  
reg\_scaled.fit(x\_train\_pp, y\_train)  
print('\n数据预处理后的模型验证集得分：{:.2f}'.format(reg\_scaled.score(x\_test\_pp, y\_test)))  
print('数据预处理后的模型训练集得分：{:.2f}'.format(reg\_scaled.score(x\_train\_pp, y\_train)))

**结果：**

文本

描述已自动生成

三、[附加题].利用Knn解决自己遇到的问题。

[解题要求同题2]

**解决红酒数据集**

import numpy as np  
import pandas as pd  
from sklearn import datasets  
# data = pd.read\_csv('iris.csv')  
# print(data.head())  
# print(data.info())  
# print(data.describe())  
  
# 加载红酒数据集  
wine = datasets.load\_wine()  
x = wine.data  
y = wine.target  
from sklearn.model\_selection import train\_test\_split  
# 将数据集划分为训练集和测试集  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=0)  
print('x\_train shape:{}'.format(x\_train.shape))  
print('x\_test shape:{}'.format(x\_test.shape))  
print('y\_train shape:{}'.format(y\_train.shape))  
print('y\_test shape:{}'.format(y\_test.shape))  
  
from sklearn.neighbors import KNeighborsRegressor  
# 使用默认参数创建K最近邻回归模型，并在训练集和测试集上进行拟合和评分  
reg = KNeighborsRegressor()  
reg.fit(x\_train, y\_train)  
print('\n默认参数的验证集得分：{:.2f}'.format(reg.score(x\_test, y\_test)))  
print('默认参数的训练集得分：{:.2f}'.format(reg.score(x\_train, y\_train)))  
  
# 创建不同参数下的K最近邻回归模型，并评估性能  
reg2 = KNeighborsRegressor(n\_neighbors=2)  
reg2.fit(x\_train, y\_train)  
print('\n模型参数n\_neighbors=2的验证集得分：{:.2f}'.format(reg2.score(x\_test, y\_test)))  
print('模型参数n\_neighbors=2的训练集得分：{:.2f}'.format(reg2.score(x\_train, y\_train)))  
  
reg10 = KNeighborsRegressor(n\_neighbors=10)  
reg10.fit(x\_train, y\_train)  
print('\n模型参数n\_neighbors=10的验证集得分：{:.2f}'.format(reg10.score(x\_test, y\_test)))  
print('模型参数n\_neighbors=10的训练集得分：{:.2f}'.format(reg10.score(x\_train, y\_train)))  
  
reg\_w = KNeighborsRegressor(weights='distance')  
reg\_w.fit(x\_train, y\_train)  
print('\n模型参数weights=distance的验证集得分：{:.2f}'.format(reg\_w.score(x\_test, y\_test)))  
print('模型参数weights=distance的训练集得分：{:.2f}'.format(reg\_w.score(x\_train, y\_train)))  
  
from sklearn.preprocessing import MinMaxScaler  
# 使用MinMaxScaler对特征进行缩放  
x\_2 = MinMaxScaler().fit\_transform(x)  
# print('\n',x\_2)  
x\_train\_pp, x\_test\_pp, y\_train, y\_test = train\_test\_split(x\_2, y, random\_state=0)  
  
# 在经过数据预处理后的数据上训练K最近邻回归模型并评估性能  
reg\_scaled = KNeighborsRegressor()  
reg\_scaled.fit(x\_train\_pp, y\_train)  
print('\n数据预处理后的模型验证集得分：{:.2f}'.format(reg\_scaled.score(x\_test\_pp, y\_test)))  
print('数据预处理后的模型训练集得分：{:.2f}'.format(reg\_scaled.score(x\_train\_pp, y\_train)))

**结果：**

文本

描述已自动生成