#获取实验1数据矩阵

import pandas as pd

data = pd.read\_csv('./实验数据/Student2.csv')

#删掉一些不必要字段

data = data.drop(['Name', 'City', 'Gender', 'Height', 'Constitution'], axis = 1)

data

#1、以课程1成绩为x轴，体能成绩为y轴，制作散点图

#1.1获取x，y轴数据

import matplotlib.pyplot as plt

x = data['C1']

y = data['Constitution\_int']

#将1.2体测数值化的数据对应成评价

plt.yticks([4, 3, 2, 1], ['$excellent$', '$good$', '$general$', '$bad$'])

#1.3制作散点图

plt.scatter(x, y)

plt.show()

#2、以5分为间隔，制作课程1的成绩直方图

#2.1让x轴以5的距离分隔开

plt.xticks(range(60, 101)[::5])

#2.2制作直方图

plt.hist(x, histtype = 'bar', bins = [60, 65, 70, 75, 80, 85, 90, 95, 100])

plt.show()

#3、归一化, 用data\_z存储归一化的数据矩阵

data\_z = pd.DataFrame()

data\_z.loc[:, 'C1'] = (data['C1'] - data['C1'].min()) / (data['C1'].max() - data['C1'].min())

data\_z.loc[:, 'C2'] = (data['C2'] - data['C2'].min()) / (data['C2'].max() - data['C2'].min())

data\_z.loc[:, 'C3'] = (data['C3'] - data['C3'].min()) / (data['C3'].max() - data['C3'].min())

data\_z.loc[:, 'C4'] = (data['C4'] - data['C4'].min()) / (data['C4'].max() - data['C4'].min())

data\_z.loc[:, 'C5'] = (data['C5'] - data['C5'].min()) / (data['C5'].max() - data['C5'].min())

data\_z.loc[:, 'C6'] = (data['C6'] - data['C6'].min()) / (data['C6'].max() - data['C6'].min())

data\_z.loc[:, 'C7'] = (data['C7'] - data['C7'].min()) / (data['C7'].max() - data['C7'].min())

data\_z.loc[:, 'C8'] = (data['C8'] - data['C8'].min()) / (data['C8'].max() - data['C8'].min())

data\_z.loc[:, 'C9'] = (data['C9'] - data['C9'].min()) / (data['C9'].max() - data['C9'].min())

data\_z.loc[:, 'Constitution'] = (data['Constitution\_int'] - data['Constitution\_int'].min()) / (data['Constitution\_int'].max() - data['Constitution\_int'].min())

data\_z

#4、计算协相关矩阵，并画出混淆矩阵

#4.1计算相关矩阵data\_corr

import numpy as np

data\_corr = pd.DataFrame(np.corrcoef(data\_z))

data\_corr

#4.2把C1作为特征值，Constitution作为目标值举例进行机器学习

#导包

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import MinMaxScaler

#划分训练集80%和测试集20%

x\_train, x\_test, y\_train, y\_test = train\_test\_split(data['C1'], data['Constitution\_int'], test\_size = 0.2)

#用归一化预处理

tra = MinMaxScaler()

x\_train = tra.fit\_transform(pd.DataFrame(x\_train))

x\_test = tra.fit\_transform(pd.DataFrame(x\_test))

#用k近邻模型进行训练

es = KNeighborsClassifier(n\_neighbors = 5)

es.fit(x\_train, y\_train.astype('int'))

#实际与预测结果相比较

print(es.predict(x\_test))#预测

print(y\_test.astype(int))#实际

#比较得分

print(es.score(x\_test, y\_test))

#将相关矩阵绝对值化

data\_corr\_abs = abs(data\_corr)

#插入对应的ID

data\_corr\_abs.loc[:, 'ID'] = data['ID']

data\_corr\_abs

#按每一列进行降序排序，提取除第2到第4个数据（第一个去掉，因为是自相关系数1）

N\_id = []

n = data['ID'].count()

for i in range(0, n) :

N\_id.append(data\_corr\_abs.sort\_values(by = i, ascending = False)['ID'][1:4])

#将得到的数据输出到txt文件

import numpy as np

np.savetxt("./实验数据/Corr\_ID.txt", N\_id, fmt = '%s', delimiter = '\t')

#输出距离每个样本最近的三个样本100x3的矩阵

pd.DataFrame(np.loadtxt("./实验数据/Corr\_ID.txt", dtype = str, encoding = 'UTF-8'), columns = ['i1', 'i2', 'i3'])