**广州大学学生实验报告**

**开课学院及实验室：**计算机科学与网络工程学院/电子信息楼  **2020 年 月 日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学 院** | 计算机科学与网络工程学院 | **年级/专业/班** | 软件182 | **姓名** | 张健鸿 | **学号** | 1806300065 |
| **实验课程名称** | **机器学习与数据挖掘实验** | | | | | **成绩** |  |
| **实验项目名称** | **实验2** | | | | | **指导老师** | 彭伟龙 |

(\*\*\*（1）报告只能为文字和图片；（2）实验项目名称不能有任何改动；（3）必须填写正确的姓名和学号；否则会影响最后成绩的统计，后果自负\*\*\*)

**一、实验目的及要求**

1. 掌握对多维数据统计、数据立方的理解

2. 掌握基本数据归一化、散点可视化、分布可视化方法

3. 按照既定格式书写实验报告

**二、实验设备与平台**

1. 实验设备：计算机；

2. 平台：Windows 10操作系统

**三、实验内容**

**题目**：基于实验一中清洗后的数据练习统计和视化操作，100个同学（样本），每个同学有11门课程的成绩（11维的向量）；那么构成了一个100x11的数据矩阵。以你擅长的语言，编程计算：

1. 请以课程1成绩为x轴，体能成绩为y轴，画出散点图。
2. 以5分为间隔，画出课程1的成绩直方图。
3. 对每门成绩进行z-score归一化，得到归一化的数据矩阵。
4. 计算出100x100的相关矩阵，并可视化出混淆矩阵。（为避免歧义，这里“协相关矩阵”进一步细化更正为100x100的相关矩阵，100为学生样本数目，视实际情况而定）
5. 根据相关矩阵，找到距离每个样本最近的三个样本，得到100x3的矩阵（每一行为对应三个样本的ID）输出到txt文件中，以\t,\n间隔

**实验实施**：

（在此详述平台，技术栈，思路，处理逻辑等等）

本实验采用经过实验一处理后的数据进行操作

1）绘制散点图，以体能测试为y轴，课程1成绩为x轴，此处继续采用实验一给出的成绩对应字典进行绘图，绘图函数如下

1. # 绘制散点图
2. **def** DrawScatter(txtData, a, b):
3. x = []
4. y = []
5. index = [i **for** i **in** range(4)]
6. con = ['bad', 'general', 'good', 'excellent']
7. plt.yticks(index, con)
8. **for** i **in** range(txtData.shape[0]):
9. x.append(txtData[i][a])
10. y.append(txtData[i][b])
11. plt.scatter(x, y)
12. plt.show()

2）绘制条形图，因为要每5分绘制一个柱状图，此处采用四舍五入的思想，成绩如果个位数小于3，归为0，如果处在4到7之间，归为5，处在8/9则十位数进一，个位归零

1. **def** DrawHist(txtData, a):
2. x = []
3. **for** i **in** range(txtData.shape[0]):
4. **if** dic\_near[str(int(txtData[i][a] % 10))] == 0:
5. x.append(int(txtData[i][a] / 10) \* 10)
6. **elif** dic\_near[str(int(txtData[i][a] % 10))] == 1:
7. x.append(int(txtData[i][a] / 10) \* 10 + 5)
8. **else**:
9. x.append((int(txtData[i][a] / 10) + 1) \* 10)
10. plt.hist(x, histtype='bar', rwidth=0.8)
11. plt.show()

3）对每门成绩归一化，并生成归一化矩阵，此处计算每一列的均值和方差，采用教师给出的z-score归一化公式，将每一个数据进行计算，得出他们的归一值，此处采用的calMean函数和calStd为实验一定义的求均值、求方差函数，这里不再重复定义

1. **def** Z\_Score(txtData):
2. Study\_Avg = CalMean(txtData, title.index('C1'), title.index('C10'))
3. Study\_Std = CalStd(
4. txtData,
5. title.index('C1'),
6. title.index('C10'),
7. Study\_Avg)
8. **for** i **in** range(len(Study\_Std)):
9. Study\_Std[i] = math.sqrt(Study\_Std[i])
10. Sport\_Avg = 0
11. **for** i **in** range(txtData.shape[0]):
12. **if** txtData[i][title.index('CONSTITUTION')] == 'bad':
13. Sport\_Avg += 50
14. **elif** txtData[i][title.index('CONSTITUTION')] == 'general':
15. Sport\_Avg += 65
16. **elif** txtData[i][title.index('CONSTITUTION')] == 'good':
17. Sport\_Avg += 80
18. **elif** txtData[i][title.index('CONSTITUTION')] == 'excellent':
19. Sport\_Avg += 90
21. Sport\_Avg = Sport\_Avg / txtData.shape[0]
23. Sport\_std = 0
24. **for** i **in** range(txtData.shape[0]):
25. **if** txtData[i][title.index('CONSTITUTION')] == 'bad':
26. Sport\_std += (50 - Sport\_Avg) \*\* 2 / txtData.shape[0]
27. **elif** txtData[i][title.index('CONSTITUTION')] == 'general':
28. Sport\_std += (65 - Sport\_Avg) \*\* 2 / txtData.shape[0]
29. **elif** txtData[i][title.index('CONSTITUTION')] == 'good':
30. Sport\_std += (80 - Sport\_Avg) \*\* 2 / txtData.shape[0]
31. **elif** txtData[i][title.index('CONSTITUTION')] == 'excellent':
32. Sport\_std += (90 - Sport\_Avg) \*\* 2 / txtData.shape[0]
33. Sport\_std = math.sqrt(Sport\_std)
34. Z = []
35. **for** i **in** range(txtData.shape[0]):
36. **for** j **in** range(title.index('C1'), title.index('CONSTITUTION') + 1):
37. **if** txtData[i][j] == 'bad':
38. Z.append((50 - Sport\_Avg) / Sport\_std)
39. **break**
40. **if** txtData[i][j] == 'general':
41. Z.append((65 - Sport\_Avg) / Sport\_std)
42. **break**
44. **if** txtData[i][j] == 'good':
45. Z.append((80 - Sport\_Avg) / Sport\_std)
46. **break**
48. **if** txtData[i][j] == 'excellent':
49. Z.append((90 - Sport\_Avg) / Sport\_std)
50. **break**
52. **if** np.isnan(txtData[i][j]):
53. Z.append(0)
54. **continue**
55. Z.append((txtData[i][j] - Study\_Avg[j - 5]) / Study\_Std[j - 5])
56. Z = np.array(Z)
57. Z = Z.reshape((txtData.shape[0], 11))
58. **return** Z

4）输出100\*100的相关矩阵，并进行可视化，此处运用求相关系数的公式，先对每一行的z值求均值和方差，然后用他们的z值与剩下的每个学生进行相关系数的运算，最后用matplot中的热力图输出混淆矩阵：

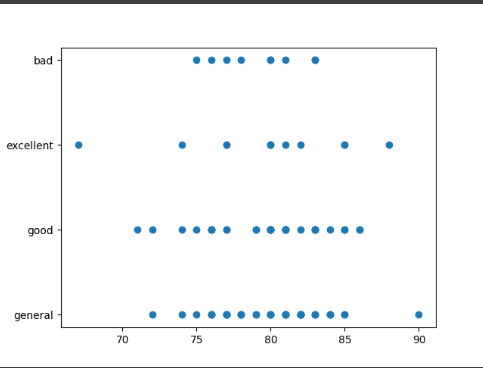
1. **def** CorretionMatrix(z):
2. CorrMartix = np.zeros((z.shape[0], z.shape[0]))
3. mean = []
4. # 计算每一行Z\_SCORE归一化矩阵的均值
5. **for** i **in** range(z.shape[0]):
6. x = 0
7. **for** j **in** range(z.shape[1]):
8. x += z[i][j]
9. mean.append((x / z.shape[1]))
10. std = []
11. # 计算每一行的方差
12. **for** i **in** range(z.shape[0]):
13. x = 0
14. **for** j **in** range(z.shape[1]):
15. x += ((z[i][j]) - mean[i])\*\*2
16. std.append(x / (z.shape[1] - 1))
17. **for** i **in** range(z.shape[0]):
19. **for** k **in** range(z.shape[0]):
20. count = 0
21. # if i==k:
22. #     CorrMartix.append(1)
23. #     continue
24. **for** j **in** range(z.shape[1]):
25. count += (z[i][j] - mean[i]) \* \
26. (z[k][j] - mean[k]) / (z.shape[1] - 1)
27. count = count / math.sqrt(std[i] \* std[k])
28. CorrMartix[i][k] = count
29. **return** CorrMartix
30. sns.heatmap(data=CorrMartix, annot=True)

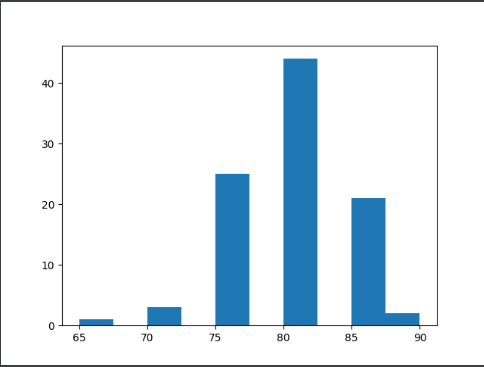
5）对输出的混淆矩阵进行相关性的排序，输出距离每个样本最近的三个样本，这里直接用numpy中的argsort进行排序，由于每行的第一个元素必然是自己本身，所以需要剔除每行的首个元素，保存为.txt格式

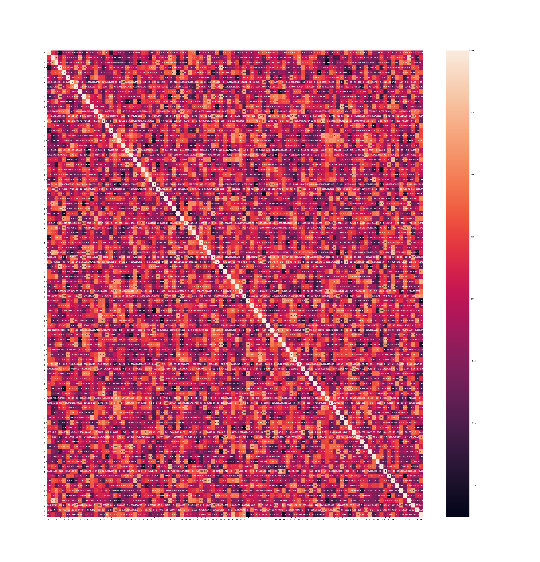
1. **for** i **in** range(CorrMartix.shape[0]):
2. final.append(CorrMartix[i].argsort()[-4:][::-1])
3. final = np.delete(final, 0, axis=1)
4. np.savetxt('result.txt', final, fmt='%d')

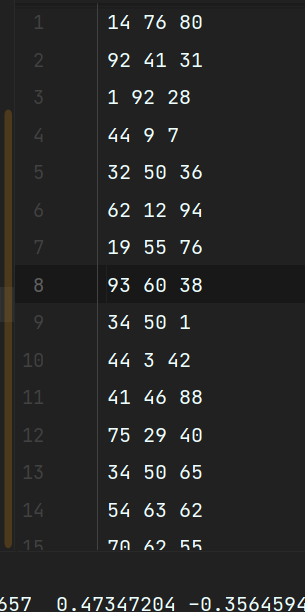
**运行结果**：

（运行结果、现象的说明与截图）









**实验总结**：

（遇到的问题，有什么感想和收货等等）

本次实验还是在实验一的基础上进行的，实验过程中理解透彻z-score的公式以及如何计算相关系数是比较关键的一步，开始因为搞错了公式，所以相关矩阵对角线上的相关性总不是为1，这很明显是存在问题的，最后经过对公式的反复推敲后，修改程序得出了正确的运算结果，本次实验加强了我对公式的理解，以及对matplot一些基础画图函数的运用，加深了我对python语句的熟练度