

《数据可视化实验》

报告

**学 号： 22009200894**

**姓 名： 王越洋**

**《数据可视化实验》报告评分表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程名称** | **数据可视化** | | | | |
| **学 号** | **22009200894** | **姓名** | **王越洋** | **成绩** |  |
| **题 目** | 实验六：时序多变量数据可视化 | | | | |
| **要 求** | （1）分析学习者答题行为日志记录，从答题分数、答题状态等多维度属性量化评估知识点掌握程度，并识别其知识体系中存在的薄弱环节。  （2）结合学习者的特征挖掘个性化学习行为模式，从多角度设计并展示学习者画像，如答题高峰时段、偏好题型、正确答题率等。  （3）不同的学习模式直接影响到学习者对知识的吸收、整合及应用能力，高效的学习模式能够促进知识的深度理解和长期记忆。请对学习模式与知识掌握程度之间的潜在关系进行建模，利用图表的形式呈现结果并简要分析。  （4）合理的题目难度应当与学习者的知识掌握程度相匹配，当学习者知识掌握水平很高但答题正确率较低时，意味着题目难度超出了其能力范围。请试着利用可视分析方法找出这些不合理的题目。  （5）结合上述分析结果，请你为题目设计者和课程管理人员提供一些宝贵的建议，以优化题库内容设置和改善教学质量，并简要说明理由。 | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **评价项目** | **评价标准** | **满分** | **得分** |
| 方案合理性 | 能够根据题目的要求，对问题进行分析，利用了一定的基础理论和科学方法来解决实际工程问题，给出了完整的解决方案，报告中理论准备充分，有充分的分析论证过程。 | **15** |  |
| 结果正确性 | 报告数据与分析详实、正确、可信；  设计、实验及测试数据的充分性和可靠性；  理论推导或建模的严密性和完整性。 | **15** |  |
| 报告水平 | 全面了解本领域的动态，并能很好地评述研究背景（文献、市场、需求等）；  报告内容涉及较为深入的基础理论知识、专业技术知识、相关学科专业知识；  具有适当的技术难度，工作量较大。 | **25** |  |
| 写作能力 | 图表清晰，语言规范，符合实验报告要求；  设计、图表、软件的规范化和标准化；  文字表达的专业性、通顺性和概念的准确性；  论文构架的系统性、逻辑性和严谨性；  引用文献的真实性、贴切性和规范性。 | **15** |  |
| 演示视频 | 视频图像和声音清晰，能结合文档更好地展现作业的完成情况。 | **15** |  |
| 使用现代工具 | 能够针对复杂工程问题，开发、选择与使用恰当的技术、资源、现代工程工具和信息技术工具，包括对复杂工程问题的预测与模拟，并能够理解其局限性。 | **15** |  |
| **总 分** | | |  |

**评阅人（签字）： 2023年 月 日**

|  |
| --- |
| **一、实验目的及要求**  本实验的主要目标是掌握层次和网络可视化和文本可视化的基本原理和方法。基于学习者的答题行为日志记录，从多个维度分析其学习行为，以量化评估学习者对知识点的掌握情况。通过答题分数、答题状态等多维度属性，我们希望识别出学习者在知识体系中存在的薄弱环节。此外，结合学习者的特征，挖掘个性化的学习行为模式，生成学习者画像，包括答题高峰时段、偏好题型、正确答题率等方面。  具体要求如下：  （1）量化知识点掌握程度：基于答题分数、答题状态等多维度数据，评估学习者对知识点的掌握情况。  （2）识别薄弱环节：识别出学习者在知识体系中存在的薄弱环节。  （3）学习者画像分析：从学习者的答题行为、偏好题型、答题高峰时段等方面，分析其学习行为模式。  （4）建模学习模式与知识掌握的关系：分析不同学习模式对知识掌握程度的影响，并通过图表展示。  （5）可视化不合理题目：通过分析，找出与学习者知识掌握程度不匹配的题目，判断题目难度设置是否合理。  （6）优化建议：基于分析结果，为题目设计者和课程管理人员提供建议，以优化题库内容并改善教学质量。  **二、实验环境配置**  1.软件环境：  操作系统：Windows 10  开发工具：Jupyter Notebook / Python IDE  Python 版本：3.8+  数据处理库：Pandas  可视化库：Pyecharts  2.数据来源：  学习者答题日志记录（包含答题状态、时间消耗、分数等）  学习者的个体信息（如年龄、学习偏好等）  题目属性（知识点、难度等）  **三、实验步骤**  1. 数据加载与预处理  从不同文件中加载学习者的答题记录、题目属性和学习者的个体信息。  合并数据集，确保所有数据整合在一起，便于后续分析。主要合并的字段有 title\_ID（题目ID） 和 student\_ID（学生ID）。  处理数据中的缺失值，将时间消耗列转换为数值，并根据需要将时间戳转换为小时等相关单位。  2. 图表设计与生成  按照实验要求设计了多个图表来量化学习者的知识掌握情况和学习行为模式，并分析知识掌握程度与学习模式之间的关系。  生成的图表包括：  （1）编程语言使用偏好：展示不同编程语言的使用频率。  使用饼图来展示不同编程语言的答题次数，反映学习者的编程语言偏好。  （2）编程语言与平均得分分析：量化不同编程语言下的答题平均得分。  使用柱状图展示学习者在不同编程语言下的得分情况，帮助了解学习者在不同编程语言下的表现。  （3）编程语言与答题耗时分析：分析不同编程语言的平均答题耗时。  使用折线图展示学习者在不同编程语言下的平均答题耗时，以便理解编程语言对答题速度的影响。  （4）知识点掌握情况：量化各知识点的平均得分。  使用柱状图展示学习者在不同知识点上的平均得分，识别出知识薄弱环节。  （5）答题活跃时段分析：展示学习者答题的高峰时段。  使用柱状图展示学习者答题的活跃时段，以便了解其学习时间分布。  （6）题型偏好分析：展示学习者对不同题型的偏好程度。  使用柱状图统计各知识点的答题次数，了解学习者的题型偏好。  （7）正确答题率分析：计算并展示各题型的正确答题率。  使用折线图展示不同知识点的正确答题率，判断学习者在哪些知识点上掌握不足。  （8）学习模式与知识掌握关系：分析答题耗时与得分之间的关系。  使用折线图展示答题耗时与平均得分的关系，以便理解学习模式对知识掌握的影响。  （9）每道题的得分与答题量关系：分析每道题的得分与答题量，识别不合理的题目。  使用气泡图展示每道题的得分与答题量，通过答题量大小与得分水平的关系，判断题目难度的合理性。  （10）年龄与答题表现分析：分析不同年龄段的答题得分情况。  使用柱状图展示不同年龄段的平均得分，了解不同年龄段学习者的表现差异。  布局：  将每两个图表并排展示在一行，以增强页面的可读性。  默认显示 SubmitRecord-Class1 的看板，用户可通过页面上的选择列表切换不同班级的看板。  **四、实验结果**    **五、源代码**   1. *# 导入必要的库* 2. import pandas as pd 3. from pyecharts.charts import Bar, Line, Pie, Scatter, Page 4. from pyecharts import options as opts 5. import os 6. *# 加载题目和学生的属性数据* 7. question\_attributes = pd.read\_csv('./数据/题目属性.csv') 8. student\_attributes = pd.read\_csv('./数据/学生属性.csv') 9. *# 获取所有班级数据文件的路径* 10. file\_paths = [ 11. './数据/Data\_SubmitRecord/SubmitRecord-Class1.csv', 12. './数据/Data\_SubmitRecord/SubmitRecord-Class2.csv', 13. './数据/Data\_SubmitRecord/SubmitRecord-Class3.csv', 14. './数据/Data\_SubmitRecord/SubmitRecord-Class4.csv', 15. './数据/Data\_SubmitRecord/SubmitRecord-Class5.csv', 16. './数据/Data\_SubmitRecord/SubmitRecord-Class6.csv', 17. './数据/Data\_SubmitRecord/SubmitRecord-Class7.csv', 18. './数据/Data\_SubmitRecord/SubmitRecord-Class8.csv', 19. './数据/Data\_SubmitRecord/SubmitRecord-Class9.csv', 20. './数据/Data\_SubmitRecord/SubmitRecord-Class10.csv', 21. './数据/Data\_SubmitRecord/SubmitRecord-Class11.csv', 22. './数据/Data\_SubmitRecord/SubmitRecord-Class12.csv', 23. './数据/Data\_SubmitRecord/SubmitRecord-Class13.csv', 24. './数据/Data\_SubmitRecord/SubmitRecord-Class14.csv', 25. './数据/Data\_SubmitRecord/SubmitRecord-Class15.csv' 26. ] 27. *# 遍历每个班级的数据文件，生成独立的 HTML 看板页面* 28. for file\_path in file\_paths: 29. submit\_records = pd.read\_csv(file\_path) 30. class\_name = os.path.splitext(os.path.basename(file\_path))[0] 32. *# 合并数据* 33. merged\_data = pd.merge(submit\_records, question\_attributes, on='title\_ID', 34. how='left') 35. merged\_data = pd.merge(merged\_data, student\_attributes, on='student\_ID', 36. how='left') 37. merged\_data['timeconsume'] = pd.to\_numeric(merged\_data['timeconsume'], 38. errors='coerce') 39. *# 定义所有图表* 40. language\_distribution = merged\_data['method'].value\_counts().reset\_index() 41. language\_distribution.columns = ['编程语言', '答题次数'] 42. pie\_language = ( 43. Pie() 44. .add("编程语言", [list(z) for z in zip(language\_distribution['编程语言'], 45. language\_distribution['答题次数'])]) 46. .set\_global\_opts(title\_opts=opts.TitleOpts(title="编程语言使用偏好")) 47. ) 48. language\_score = merged\_data.groupby('method')['score\_x'].mean().reset\_index() 49. bar\_language\_score = ( 50. Bar() 51. .add\_xaxis(language\_score['method'].tolist()) 52. .add\_yaxis("平均得分", language\_score['score\_x'].tolist()) 53. .set\_global\_opts(title\_opts=opts.TitleOpts(title="编程语言与平均得分")) 54. ) 55. language\_timeconsume = merged\_data.dropna(subset=['timeconsume']). 56. groupby('method')['timeconsume'].mean().reset\_index() 57. line\_language\_timeconsume = ( 58. Line() 59. .add\_xaxis(language\_timeconsume['method'].tolist()) 60. .add\_yaxis("平均耗时 (秒)", language\_timeconsume['timeconsume'].tolist()) 61. .set\_global\_opts(title\_opts=opts.TitleOpts(title="编程语言与答题耗时")) 62. ) 63. knowledge\_mastery = merged\_data.groupby('knowledge')['score\_x'].mean(). 64. reset\_index() 65. bar\_knowledge = ( 66. Bar() 67. .add\_xaxis(knowledge\_mastery['knowledge'].tolist()) 68. .add\_yaxis("平均得分", knowledge\_mastery['score\_x'].tolist()) 69. .set\_global\_opts(title\_opts=opts.TitleOpts(title="知识点掌握情况")) 70. ) 71. merged\_data['hour'] = pd.to\_datetime(merged\_data['time'], unit='s').dt.hour 72. hourly\_distribution = merged\_data.groupby('hour').size().reset\_index 73. (name='答题次数') 74. bar\_time = ( 75. Bar() 76. .add\_xaxis([f"{i}点" for i in hourly\_distribution['hour']]) 77. .add\_yaxis("答题次数", hourly\_distribution['答题次数'].tolist()) 78. .set\_global\_opts(title\_opts=opts.TitleOpts(title="答题活跃时段分布")) 79. ) 80. *# 6. 偏好题型分析* 81. question\_type\_distribution = merged\_data['knowledge'].value\_counts(). 82. reset\_index() 83. question\_type\_distribution.columns = ['题型', '答题次数'] 84. bar\_type = ( 85. Bar() 86. .add\_xaxis(question\_type\_distribution['题型'].tolist()) 87. .add\_yaxis("答题次数", question\_type\_distribution['答题次数'].tolist()) 88. .set\_global\_opts( 89. title\_opts=opts.TitleOpts(title="题型偏好分析"), 90. xaxis\_opts=opts.AxisOpts(name="题型"), 91. yaxis\_opts=opts.AxisOpts(name="答题次数") 92. ) 93. ) 94. correct\_rate = merged\_data.groupby('knowledge')['state'].apply( 95. lambda x: (x == 'Absolutely\_Correct').mean() 96. ).reset\_index() 97. line\_correct\_rate = ( 98. Line() 99. .add\_xaxis(correct\_rate['knowledge'].tolist()) 100. .add\_yaxis("正确答题率", (correct\_rate['state'] \* 100).tolist()) 101. .set\_global\_opts(title\_opts=opts.TitleOpts(title="正确答题率分析")) 102. ) 103. learning\_behavior = merged\_data.groupby('timeconsume')['score\_x']. 104. mean().reset\_index() 105. line\_timeconsume = ( 106. Line() 107. .add\_xaxis(learning\_behavior['timeconsume'].tolist()) 108. .add\_yaxis("平均得分", learning\_behavior['score\_x'].tolist()) 109. .set\_global\_opts(title\_opts=opts.TitleOpts(title= 110. "答题耗时与知识掌握程度关系")) 111. ) 112. *# 9. 每道题的得分与答题量关系（气泡图）* 113. title\_score\_data = merged\_data.groupby('title\_ID').agg( 114. avg\_score=('score\_x', 'mean'), 115. count=('score\_x', 'size') 116. ).reset\_index() 117. title\_list = title\_score\_data['title\_ID'].tolist() 118. avg\_score\_list = title\_score\_data['avg\_score'].tolist() 119. count\_list = title\_score\_data['count'].tolist() 120. scatter = ( 121. Scatter() 122. .add\_xaxis(title\_list) 123. .add\_yaxis( 124. "Average Score", 125. avg\_score\_list, 126. symbol\_size=[min(50, max(10, c / 2)) for c in count\_list], 127. *# 根据答题量调整气泡大小* 128. itemstyle\_opts=opts.ItemStyleOpts(opacity=0.7)  *# 设置气泡透明度* 129. ) 130. .set\_global\_opts( 131. title\_opts=opts.TitleOpts(title="每道题的得分与答题量关系"), 132. xaxis\_opts=opts.AxisOpts(name="题目ID", axislabel\_opts= 133. {"rotate": 45}), 134. yaxis\_opts=opts.AxisOpts(name="平均得分", min\_=0, max\_=4), 135. *# 假设得分满分为4* 136. tooltip\_opts=opts.TooltipOpts(trigger="item", formatter=" 137. {b}: 平均得分 {c}"), 138. legend\_opts=opts.LegendOpts(pos\_top="5%", pos\_right="5%"), 139. *# 设置图例位置为右上角* 140. visualmap\_opts=opts.VisualMapOpts( 141. type\_="size", 142. max\_=50, 143. min\_=10, 144. dimension=1, 145. pos\_right="5%",  *# 右对齐* 146. pos\_top="15%",   *# 图例下方* 147. ) 148. ) 149. ) 150. age\_score = merged\_data.groupby('age')['score\_x'].mean().reset\_index() 151. bar\_age\_score = ( 152. Bar() 153. .add\_xaxis(age\_score['age'].astype(str).tolist()) 154. .add\_yaxis("平均得分", age\_score['score\_x'].tolist()) 155. .set\_global\_opts(title\_opts=opts.TitleOpts(title="年龄与答题表现")) 156. ) 157. *# 加入到页面布局中* 158. charts = [ 159. pie\_language, bar\_language\_score, line\_language\_timeconsume, 160. bar\_knowledge, 161. bar\_time, bar\_type, line\_correct\_rate, line\_timeconsume, 162. scatter, bar\_age\_score 163. ] 165. *# 创建 HTML 页面，并在表格布局中两两排列图表* 166. with open(f"./数据/{class\_name}\_dashboard.html", "w", encoding="utf-8") 167. as f: 168. f.write("<html><head><title>班级数据看板</title></head><body> 169. <h1>班级数据看板</h1>") 170. f.write("<table style='width:100%'>") 172. for i in range(0, len(charts), 2): 173. f.write("<tr>") 174. f.write("<td style='width:50%'>") 175. f.write(charts[i].render\_embed())  *# 渲染第一个图表* 176. f.write("</td>") 178. if i + 1 < len(charts): 179. f.write("<td style='width:50%'>") 180. f.write(charts[i + 1].render\_embed())  *# 渲染第二个图表* 181. f.write("</td>") 182. f.write("</tr>") 184. f.write("</table></body></html>") 185. print(f"{class\_name}的看板已生成为 ./数据/{class\_name}\_dashboard.html") 186. *# 创建主页面用于选择班级* 187. with open("all\_classes\_dashboard.html", "w", encoding="utf-8") as f: 188. f.write(""" 189. <html> 190. <head> 191. <title>所有班级数据看板</title> 192. <script> 193. function showClassBoard(className) { 194. const iframe = document.getElementById("classIframe"); 195. iframe.src = "./数据/" + className + "\_dashboard.html"; 196. } 197. </script> 198. </head> 199. <body> 200. <h1>所有班级数据看板</h1> 201. <label for="classSelect">选择班级:</label> 202. <select id="classSelect" onchange="showClassBoard(this.value)"> 203. """) 205. for i, file\_path in enumerate(file\_paths): 206. class\_name = os.path.splitext(os.path.basename(file\_path))[0] 207. selected = "selected" if i == 0 else "" 208. f.write(f'<option value="{class\_name}" {selected}>{class\_name} 209. </option>') 211. f.write(""" 212. </select> 213. <iframe id="classIframe" src="./数据/SubmitRecord-Class1\_dashboard.html" style="width: 100%; height: 800px; border: none;"> 214. </iframe> 215. </body> 216. </html> 217. """) 218. print("所有班级的主页面已生成")   **六、问题回答**  1.分析学习者答题行为日志记录，从答题分数、答题状态等多维度属性量化评估知识点掌握程度，并识别其知识体系中存在的薄弱环节。  知识点r83g掌握较差，  2.结合学习者的特征挖掘个性化学习行为模式，从多角度设计并展示学习者画像，如答题高峰时段、偏好题型、正确答题率等。  统计时段，12~14点处于高峰期，学生喜欢在下午做作业，统计大体次数，最受欢迎的题型为m3D1v，s8V2f题型答题次数最少，  3.不同的学习模式直接影响到学习者对知识的吸收、整合及应用能力，高效的学习模式能够促进知识的深度理解和长期记忆。请对学习模式与知识掌握程度之间的潜在关系进行建模，利  用图表的形式呈现结果并简要分析。  分析正确率和答题耗时与知识掌握程度的关系，答题耗时在50~122之间的答题时间适宜，得分较高，各题目正确率在各个班级表现情况相近，g7R2j、r8S3g、t5V9e正确率较低。同时可以发现答题耗时较快也存在得分较多的学生，证明对于优秀学生来说快节奏的学习比较适宜，但对于大部分学生应该适当放慢节奏。  4.合理的题目难度应当与学习者的知识掌握程度相匹配，当学习者知识掌握水平很高但答题正确率较低时，意味着题目难度超出了其能力范围。请试着利用可视分析方法找出这些不合理的题目。  用气泡图展示每道题的得分与答题量关系，存在得分规律偏离正常标准的题目（如图）  5.结合上述分析结果，请你为题目设计者和课程管理人员提供一些宝贵的建议，以优化题库内容设置和改善教学质量，并简要说明理由。  （1）知识点薄弱环节：建议针对掌握度较低的知识点进行强化练习或额外辅导。  （2）学习模式优化：在得分与答题耗时的关系中，合理的思考时间能够提高答题效果。因此，建议设置一些具有挑战性的题目，鼓励学习者深入思考，以提升知识掌握的深度。  （3）题库优化：针对答题量大但得分低的题目，建议题目设计者适当降低其难度，或在课程中增加相关知识点的讲解。  （4）多样化题型设计：结合学习者的偏好，丰富题型的设计，使题库更加有趣且符合学习者需求。  （5）合理分布学习内容：根据活跃时段分析的结果，考虑在不同时间段安排不同难度的内容，以提升学习效率。 |