

《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** | 实验五：高维非空间数据可视化 | | | | |
| **要 求** | Ⅱ 车辆驾驶画像可视分析  面向社会车辆驾驶员，构建多种大数据标签，对其驾驶行为进行画像分析。建议构建评分系统或其他量化方式，对画像进行评级评定，进行可视化展示并实现画像群体的可视分析。 | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **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**  本实验的目的是构建一套基于大数据的车辆驾驶员驾驶行为画像系统，面向社会车辆驾驶员，针对不同种类车辆的数据进行分析和可视化。通过建立模型对驾驶行为进行画像分析，结合道路数据处理和不同种类车辆的行为数据，实现驾驶行为的量化和可视化展示。系统最终需实现画像群体的可视分析。  **实验要求**  （1）数据处理：对道路数据和车辆行为数据进行预处理和可视化。  （2）驾驶行为画像：为不同种类车辆构建多维标签数据，通过聚类、评分系统或量化方式对驾驶行为画像进行评级。  （3）可视化：将驾驶行为画像数据进行可视化展示，通过页面切换和选择，实现对多张Excel表格的不同分析结果页面的切换展示。  （4）模型方法：选择适合的算法和模型对驾驶行为数据进行分类聚类分析，并对道路和车辆行为进行处理、分析和展示。  **二、实验环境配置**  操作系统：Windows 10  编程语言：Python 3.x  主要依赖库：  Pandas：用于数据读取和数据处理。  Geopandas：用于处理地理数据，支持对道路数据的可视化处理。  Pyecharts：用于动态和静态的图表生成，支持多图表的页面组合和选择。  Matplotlib：用于生成聚类图并保存为图像。  Scikit-learn：用于K-means聚类分析。  Jupyter Notebook / IDE：Anaconda配置的Jupyter Notebook 或者任意兼容Python代码的IDE。  **三、实验步骤**  1. 数据读取与预处理  （1）道路数据：使用geopandas读取GeoJSON格式的道路数据文件，并将其可视化。道路数据包含每条路段的地理信息，以LineString或Polygon的格式表示。处理时，将每个坐标点提取出来并在地图上展示，以还原实际道路网络。  （2）车辆数据：读取每个Excel文件中的车辆行为数据，主要字段包括速度、方向偏差、车辆种类、坐标点等。将每种类型的车辆行为数据单独处理并展示，以便后续进行聚类分析和评分系统的构建。  2. 数据聚类及驾驶行为画像构建  （1）构建驾驶行为标签：将车辆行为数据基于速度、方向偏差等特征构建标签，为每种车辆生成驾驶行为画像。  （2）K-means聚类：  使用Scikit-learn中的KMeans对数据进行聚类，聚类数设定为4。聚类标签用于将驾驶行为分成不同的类型。  聚类特征选择了速度和方向偏差，聚类后可将驾驶行为分为四类（簇），生成驾驶行为画像。  为每个聚类类别分配一个标签，例如：正常、激进、谨慎等驾驶风格，作为驾驶行为画像标签。  （3）评分系统的构建：  在聚类的基础上，构建一个评分系统，评分基于车辆在各个特征上的表现（例如速度和方向偏差的稳定性和平均值）。  对不同驾驶风格进行评分：比如，较高的速度和较大的方向偏差可能会被标记为“激进驾驶”，评分较低；而稳定的速度和较小的方向偏差则被认为是“谨慎驾驶”，评分较高。  3. 数据可视化及页面展示  （1）静态聚类图的生成：  使用Matplotlib生成聚类结果的散点图，将每个聚类类别以不同颜色标识，并保存为图片。  每个聚类结果图包含四个聚类类别的分布点和聚类中心，以不同颜色和标记展示。  （2）Pyecharts可视化展示：  **条形图：**展示每种车辆类型的数量分布，用于了解数据中各类车辆的占比。  **散点图：**使用道路数据和车辆坐标点数据绘制道路网络图，结合车辆行为数据展示每种车辆类型的行驶路径和分布。  **折线图：**展示每种车辆类型在不同ID（数据点）下的速度变化情况，帮助理解车辆的行驶速度随时间变化的趋势。  **聚类图嵌入**：将生成的聚类结果图片以Base64编码的方式嵌入到HTML文件中，使得每个页面包含车辆行为数据的聚类结果图。  （3）多页面切换功能：  使用HTML和JavaScript实现页面切换。主页面提供下拉菜单，用户可以选择想要查看的Excel文件，每个选择对应一个分析结果页面。  选中不同的页面后，会在同一个HTML容器中动态加载不同的图表展示，以实现对多张表格的可视化结果切换。  **四、实验结果**  1. 数据预处理与聚类结果  （1）道路数据展示：道路数据成功读取并展示，道路的多边形和线条在地图上清晰可见，重现了实际的道路网络结构。  （2）车辆行为数据展示：Excel数据读取成功，并通过条形图、散点图和折线图展示了每种车辆类型的数量、行驶路径以及速度变化情况。  （3）K-means聚类结果：聚类分析成功运行，将车辆的驾驶行为分成了四类，分别标记为不同的驾驶风格（如：正常、激进、谨慎等）。聚类结果通过散点图展示，每个类别以不同的颜色和标记展示在聚类图中，并保存为静态图像嵌入到页面中。  2. 可视化展示与多页面切换  （1）多页面切换：通过HTML页面的下拉菜单实现了多页面的展示切换功能，用户可以选择不同的Excel文件对应的页面进行浏览。  （2）图表展示效果：每个页面包含四个图表展示：  条形图展示每种车辆的数量。  散点图展示了道路数据和车辆行驶轨迹。  折线图展示车辆的速度变化。  聚类图展示车辆驾驶行为的聚类分析结果。  （3）页面切换响应速度：切换不同页面时，图表能够快速加载，展示不同Excel文件的分析结果。  3. 驾驶行为画像分析  （1）驾驶行为评分系统：基于聚类结果的评分系统有效区分了不同驾驶行为。每个类别的车辆根据其聚类标签获得了相应的评分，便于对车辆驾驶风格进行定量化评定。  （2）画像群体的可视分析：通过多页面切换和图表展示，用户能够直观地观察不同车辆类型的驾驶行为特征，并通过聚类结果了解各类驾驶行为的群体分布。    **五、源代码**   1. *#绘图* 2. import os 3. import base64 4. from io import BytesIO 5. import pandas as pd 6. import geopandas as gpd 7. from pyecharts.charts import Bar, Scatter, Line, Grid 8. from pyecharts import options as opts 9. import matplotlib.pyplot as plt 10. from sklearn.cluster import KMeans 11. import os 12. os.environ["OMP\_NUM\_THREADS"] = "1"  *# 将线程数限制为 1，避免内存泄漏* 13. *# 设置中文字体* 14. plt.rcParams['font.sans-serif'] = ['SimHei'] 15. plt.rcParams['axes.unicode\_minus'] = False 16. *# 文件路径* 17. geojson\_file\_path = './车辆驾驶画像可视分析/road2-12-9road/boundaryroad\_with9road 18. .geojson' 19. excel\_files = ['./车辆驾驶画像可视分析/合并结果/20230413\_0000 - 0010.xlsx', 20. './车辆驾驶画像可视分析/合并结果/20230413\_0001 - 0011.xlsx', 21. './车辆驾驶画像可视分析/合并结果/20230413\_0003 - 0013.xlsx', 22. './车辆驾驶画像可视分析/合并结果/20230413\_0004 - 0014.xlsx', 23. './车辆驾驶画像可视分析/合并结果/20230413\_0010 - 0020.xlsx', 24. './车辆驾驶画像可视分析/合并结果/20230413\_0011 - 0021.xlsx', 25. './车辆驾驶画像可视分析/合并结果/20230413\_0013 - 0023.xlsx', 26. './车辆驾驶画像可视分析/合并结果/20230413\_0014 - 0024.xlsx', 27. './车辆驾驶画像可视分析/合并结果/20230413\_0020 - 0030.xlsx', 28. './车辆驾驶画像可视分析/合并结果/20230413\_0021 - 0031.xlsx', 29. './车辆驾驶画像可视分析/合并结果/20230413\_0023 - 0033.xlsx', 30. './车辆驾驶画像可视分析/合并结果/20230413\_0024 - 0034.xlsx', 31. './车辆驾驶画像可视分析/合并结果/20230413\_0030 - 0040.xlsx', 32. './车辆驾驶画像可视分析/合并结果/20230413\_0031 - 0041.xlsx', 33. './车辆驾驶画像可视分析/合并结果/20230413\_0033 - 0043.xlsx', 34. './车辆驾驶画像可视分析/合并结果/20230413\_0034 - 0044.xlsx', 35. './车辆驾驶画像可视分析/合并结果/20230413\_0041 - 0051.xlsx', 36. './车辆驾驶画像可视分析/合并结果/20230413\_0040 - 0050.xlsx', 37. './车辆驾驶画像可视分析/合并结果/20230413\_0043 - 0053.xlsx', 38. './车辆驾驶画像可视分析/合并结果/20230413\_0044 - 0054.xlsx', 39. './车辆驾驶画像可视分析/合并结果/20230413\_0050 - 0100.xlsx', 40. './车辆驾驶画像可视分析/合并结果/20230413\_0051 - 0101.xlsx', 41. './车辆驾驶画像可视分析/合并结果/20230413\_0053 - 0103.xlsx', 42. './车辆驾驶画像可视分析/合并结果/20230413\_0054 - 0104.xlsx' 43. ] 44. *# 颜色配置* 45. colors = ["blue", "orange", "green", "red"] 46. *# 类型映射字典* 47. type\_mapping = { 48. 0: "未识别", 1: "小型车辆", 2: "行人", 49. 3: "非机动车", 4: "卡车", 5: "厢式货车", 50. 6: "客车", 7: "静态物体", 8: "路牙", 51. 9: "锥桶", 10: "手推车", 11: "信号灯", 52. 12: "闸机" 53. } 54. *# 主HTML文件模板* 55. main\_html = """ 56. <!DOCTYPE html> 57. <html> 58. <head> 59. <title>Excel Analysis Selector</title> 60. <style> 61. body {{ font-family: Arial, sans-serif; }} 62. .container {{ text-align: center; }} 63. .chart-container {{ margin: auto; width: 90%; }} 64. select {{ font-size: 16px; padding: 8px; }} 65. </style> 66. </head> 67. <body> 68. <div class="container"> 69. <h1>选择时间片</h1> 70. <select id="pageSelector" onchange="selectPage()"> 71. <option value="" disabled selected>选择一个待分析的时间段</option> 72. {options} 73. </select> 74. <div class="chart-container" id="chartContainer"> 75. <!-- Placeholder for charts --> 76. </div> 77. </div> 78. <script> 79. function selectPage() {{ 80. let selector = document.getElementById("pageSelector"); 81. let selectedValue = selector.value; 82. document.getElementById("chartContainer").innerHTML = 83. `<iframe src="${{selectedValue}}" width="100%" height="1000px" frameborder="0"> 84. </iframe>`; 85. }} 86. </script> 87. </body> 88. </html> 89. """ 90. *# HTML页面选项* 91. options\_html = "" 92. *# 循环处理每个 Excel 文件* 93. for idx, excel\_file\_path in enumerate(excel\_files): 94. data = pd.read\_excel(excel\_file\_path, sheet\_name='Sheet1') 95. file\_name = os.path.basename(excel\_file\_path) 96. *# 生成聚类图并转换为Base64编码* 97. data\_for\_clustering = data[['speed', 'orientation\_heading\_diff']].dropna() 98. n\_clusters = 4 99. kmeans = KMeans(n\_clusters=n\_clusters, random\_state=0) 100. clusters = kmeans.fit\_predict(data\_for\_clustering) 101. data\_for\_clustering['cluster'] = clusters 102. fig, ax = plt.subplots(figsize=(6, 4)) 103. for cluster\_num in range(n\_clusters): 104. cluster\_data = data\_for\_clustering[data\_for\_clustering['cluster'] 105. == cluster\_num] 106. ax.scatter(cluster\_data['speed'], cluster\_data 107. ['orientation\_heading\_diff'], color=colors[cluster\_num], 108. label=f'簇 {cluster\_num}', s=10) 109. *# 绘制聚类中心* 110. centroids = kmeans.cluster\_centers\_ 111. ax.scatter(centroids[:, 0], centroids[:, 1], s=100, c='black', 112. marker='x', label='中心点') 113. ax.set\_xlabel('速度') 114. ax.set\_ylabel('方向偏差') 115. ax.set\_title('速度与方向偏差聚类分析') 116. ax.legend() 117. *# 将图像转换为Base64编码* 118. buffer = BytesIO() 119. plt.savefig(buffer, format="png") 120. plt.close(fig) 121. img\_base64 = base64.b64encode(buffer.getvalue()).decode() 122. img\_html = f'<div style="position:absolute; left:60%; bottom:15%; 123. transform: translateX(-50%);"><img src="data:image/png;base64,{img\_base64}" width="500" height="330"> 124. </div>' 125. *# ======= 生成pyecharts的其他图表 =======* 126. *# 图表1：不同车种数量的条形统计图* 127. type\_counts = data['type'].value\_counts() 128. bar\_chart = ( 129. Bar() 130. .add\_xaxis([type\_mapping.get(t, f"未知类型 {t}") for t in type\_counts. 131. index]) 132. .add\_yaxis("车种数量", type\_counts.values.tolist(), label\_opts=opts.LabelOpts(is\_show=False)) 133. .set\_global\_opts( 134. xaxis\_opts=opts.AxisOpts(name="车种类型"), 135. yaxis\_opts=opts.AxisOpts(name="数量"), 136. legend\_opts=opts.LegendOpts(pos\_top="2%", pos\_left="center", 137. orient="horizontal") 138. ) 139. ) 140. *# 图表2：道路与坐标点图（散点图）* 141. road\_data = gpd.read\_file(geojson\_file\_path) 142. scatter\_chart = Scatter() 143. scatter\_chart.set\_global\_opts( 144. xaxis\_opts=opts.AxisOpts(type\_="value", name="X坐标"), 145. yaxis\_opts=opts.AxisOpts(type\_="value", name="Y坐标") 146. ) 147. for \_, row in road\_data.iterrows(): 148. geometry = row.geometry 149. if geometry.geom\_type == 'LineString': 150. coordinates = list(geometry.coords) 151. elif geometry.geom\_type == 'Polygon': 152. coordinates = list(geometry.exterior.coords) 153. else: 154. continue 155. scatter\_chart.add\_xaxis([coord[0] for coord in coordinates]) 156. scatter\_chart.add\_yaxis("道路", [coord[1] for coord in coordinates], 157. symbol\_size=0.2, color="green", 158. label\_opts=opts.LabelOpts(is\_show=False)) 159. for t, group in data.groupby('type'): 160. scatter\_data = group[['x\_coord', 'y\_coord']].values.tolist() 161. color = colors[t % len(colors)] 162. scatter\_chart.add\_xaxis([coord[0] for coord in scatter\_data]) 163. scatter\_chart.add\_yaxis(type\_mapping.get(t, f"未知类型 {t}"), [coord[1] for coord in scatter\_data], symbol\_size=1.5, color=color, 164. label\_opts=opts.LabelOpts(is\_show=False)) 165. *# 图表3：速度统计图* 166. line\_chart = Line() 167. line\_chart.add\_xaxis(data.index.tolist()) 168. for t, group in data.groupby('type'): 169. color = colors[t % len(colors)] 170. line\_chart.add\_yaxis(type\_mapping.get(t, f"未知类型 {t}"), group['speed'].tolist(), is\_smooth=True, color=color, symbol="circle", 171. symbol\_size=3) 172. line\_chart.set\_global\_opts( 173. xaxis\_opts=opts.AxisOpts(name="ID", axislabel\_opts=opts.LabelOpts 174. (is\_show=False)), 175. yaxis\_opts=opts.AxisOpts(name="速度"), 176. legend\_opts=opts.LegendOpts(is\_show=False) 177. ) 178. *# 使用 Grid 布局，将所有图表添加到同一页面* 179. grid = Grid(init\_opts=opts.InitOpts(width="1200px", height="1000px")) 180. grid.add(bar\_chart, grid\_opts=opts.GridOpts(pos\_left="5%", 181. pos\_right="60%", 182. pos\_top="12%", pos\_bottom="55%")) 183. grid.add(scatter\_chart, grid\_opts=opts.GridOpts(pos\_left="60%", 184. pos\_right="5%", pos\_top="12%", pos\_bottom="55%")) 185. grid.add(line\_chart, grid\_opts=opts.GridOpts(pos\_left="5%", 186. pos\_right="60%", pos\_top="55%", pos\_bottom="5%")) 187. *# 输出单独的 HTML 文件* 188. single\_output\_path = f"./result/分表\_{idx + 1}.html" 189. grid.render(single\_output\_path) 190. *# 将页面选项添加到主HTML文件的下拉菜单* 191. options\_html += f'<option value="{single\_output\_path}"> 192. {file\_name}</option>' 193. *# 在每个HTML文件的底部插入聚类图HTML* 194. with open(single\_output\_path, "r+", encoding="utf-8") as file: 195. html\_content = file.read() 196. html\_content = html\_content.replace("</body>", f"{img\_html}</body>") 197. file.seek(0) 198. file.write(html\_content) 199. file.truncate() 200. *# 创建主HTML页面* 201. main\_html\_content = main\_html.format(options=options\_html) 202. with open("主界面.html", "w", encoding="utf-8") as main\_file: 203. main\_file.write(main\_html\_content) 204. print("主选择页面生成完成") |