

**实验报告册**

|  |  |
| --- | --- |
| 学年学期： | 2023 -2024学年 第1学期 |
| 课程名称： | 数据工程综合实践 |
| 学生学院： | 理学院 |
| 专业班级： | 11082303 |
| 学生学号： | 2023213851 |
| 学生姓名： | 李昌峻 |
| 联系电话： | 18225098300 |

**重庆邮电大学教务处制**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学生姓名 | 李昌峻 | 学生学号 | 2023213851 | 专业班级 | 11082303 |
| 校内指导教师 | 王进 | | | |
| 实验日期 | 2023.12 | 实验地点 | 综合实验楼B501/B502 | |
| 实验名称 | 数据工程综合实践-任务一 | | | |
| 评阅人签字 | 王进 | | | 成绩 |  |
| **一、实验目的和要求** | | | | |
| 1. 完成 Python 机器学习库与未集成算法的安装   2. 了解机器学习基本概念  3. 学习基础、中级 Python 编程  4. 学习使用第三方 Python 库  5. 了解O2O 优惠券使用预测算法基本框架  6. 完成O2O 优惠券使用预测项目解读  7. 学习数据分析与预处理方法  8. 学习数据划分与打标方法  9. 完成O2O 优惠券使用预测 Baseline | | | | |
| **二、实验设备** | | | | |
| 1、操作系统：Windows  2、基本硬件配置要求：处理器Intel i3、内存8G；  3、Python及其组件版本：  Python 3.6.5  NumPy 1.14.3  Pandas 0.23.0  Scikit-learn 0.19.1  XGBoost 0.6  LightGBM 2.1.0 | | | | |
| **三、实验内容**  **作业（二.3）**         1. 关键步骤：   1、通过rand（）函数随机生成5行10列的数字，并通过DataFrame排列。  2、通过在Pandas官网学习（<http://pandas.pydata.org/pandas-docs/stable/getting_started/intro_tutorials/04_plotting.html#min-tut-04-plotting>），了解到了pandas中自带画图功能的使用，并根据教程实现了柱状图、散点图、箱型图、面积图、六边形图的绘制   1. 代码实现：   **import** numpy as np  **import** pandas as pd  **import** matplotlib.pyplot as plt    df = np.random.rand(5, 10)  data = pd.DataFrame(df, columns=[i **for** i **in** range(1, 11)],    index=[j **for** j **in** range(1, 6)])  **print**(data)    # 画图  # 柱状图  data.plot.bar()  # 散点图  data.plot.scatter(x=1, y=2)  # 箱型图  data.plot.box()  # 面积图  data.plot(kind='area')  # 六边形图  data.plot.hexbin(x=1,y=1,gridsize=20)  plt.show()   1. 图形   1、绘制柱状图  data.plot.bar()    2、绘制散点图  data.plot.scatter(x=1, y=2)    3、绘制箱型图  data.plot.box()    4、绘制面积图  data.plot(kind='area')    5、绘制六边形图  data.plot.hexbin(x=1,y=1,gridsize=20)    **实验二 Kmeans算法**   1. 基本思路：   通过学习Kmeans算法，知道了Kmeans算法是一种经典的聚类算法，它以距离度量为基础，根据各个聚类中的对象特征的平均值进行聚类。以下是K-means算法的基本步骤：   1. 随机选择K个对象作为初始的聚类中心。 2. 对数据集中的每个对象，根据其与各个聚类中心的距离，将其分配到最近的聚类中心所在的簇中。 3. 重新计算每个簇的质心（即簇中所有对象的平均值）。 4. 重复步骤2和3，直到聚类中心不再发生变化 5. 实验步骤   (一)、核心步骤：  1、引入需要用到的库  2、随机初始化中心：  在经历循环遍历之前，需要随机找一个点作为中心点，在对其进行距离的计算，才能继续遍历。  使用 np.random.permutation 函数对数据的索引进行随机排列。这样，我们可以确保在选择前 K 个数据点作为质心时，进行随机排序。然后从随机排列后的索引中选择前 K 个数据点，并将它们赋值给 centroids。     1. 欧氏距离计算：   使用 NumPy 创建了一个形状为 (len(data), K) 的全零矩阵，用于存储每个数据点到每个质心的距离。使用 np.argmin 函数找到每个数据点到其最近质心的索引。axis=1沿着列方向寻找最小值。     1. 找到cluster，进行分类   使用 NumPy 创建了一个形状与数据集相同的全零数组，用于存储每个数据点的聚类标签。调用了前面定义的 get\_distance 函数，以计算数据集中每个点到每个质心的距离，并返回每个点到其最近质心的距离和索引。返回一个cluster数组。     1. 重新计算中心：   遍历每个簇（通过 j表示），获取该簇中所有数据点的索引，然后计算这些数据点的坐标之和，并除以数据点的数量，得到新的质心坐标。最后，返回计算得到的新质心坐标数组。、     1. Kmenas函数主体   传入data，K，max\_train，调用上述函数，进行聚类。    （二）、完整代码：   1. import numpy as np 2. import matplotlib.pyplot as plt 3. import pandas as pd 4. *# 随机初始化数据中心* 5. def random\_init\_centroids(data, K): 6. num = data.shape[0] 7. parts = np.random.permutation(num) 8. centroids = data[parts[:K], :] 9. return centroids 10. *# 获得欧氏距离* 11. def get\_distance(data, centroids, K): 12. distance = np.zeros((len(data), K))  *# 初始化距离矩阵* 13. for i in range(K): 14. distance[:, i] = np.sqrt(np.sum(np.square(data - 15. centroids[i]),axis=1)) 16. *# 计算每个点到每个质心的距离* 17. min\_distance = np.argmin(distance, axis=1) 18. *# 找到最小距离的质心索引* 19. return distance, min\_distance 20. *# 样本分类* 21. def classify\_cluster(data, centroids, K): 22. cluster = np.zeros(len(data))  *# 初始化聚类标签数组* 23. distance, min\_distance = get\_distance(data, centroids, K) 24. *# 计算距离和最小距离* 25. cluster = min\_distance  *# 将最小距离的质心索引赋值给聚类标签数组* 26. return cluster 27. *# 重新计算中心* 28. def new\_centroids(data, K, cluster, centroids): 29. for j in range(K): 30. index = (np.where(cluster == j))[0]  *# 获取簇中所有数据点的索引* 31. *# 总和除以个数得到均值（样本中心）* 32. centroids[j] = np.sum(data[index], axis=0) / len(index) 33. *# 计算新的质心坐标* 34. return centroids 35. *# K-means算法主要函数* 36. def KMeans\_train(data, K, max\_train):  *# max\_train 最大迭代训练次数* 37. *# 随机初始化质心坐标* 38. initial\_centroids = random\_init\_centroids(data, K) 39. for i in range(max\_train): 40. cluster = classify\_cluster(data, initial\_centroids, K) 41. *# 分类和计算距离* 42. initial\_centroids = new\_centroids(data, K, cluster, 43. initial\_centroids)  *# 重新计算质心坐标* 44. return cluster, initial\_centroids, K   K值确认代码（会引用到上述kmeans代码的函数，因此两者是连接在一起的）：   1. *# 读取CSV文件* 2. df = pd.read\_csv('') 3. *#数据清洗* 4. df = df.select\_dtypes(exclude=['ABC']) 5. df.to\_csv('..', index=False) 6. *# 降维* 7. from sklearn.manifold import TSNE 8. tsne=TSNE() 9. data=tsne.fit\_transform(data) 10. print(data.shape) 11. max\_train=500 *# 自定义* 12. def KMeans\_train\_with\_K(data, max\_K, max\_train): 13. distortions = []  *# 代替SSE，用于存储每个 K 对应的畸变程度* 14. for K in range(1, max\_K + 1): 15. initial\_centroids = random\_init\_centroids(data, K) 16. for i in range(max\_train): 17. cluster = classify\_cluster(data, initial\_centroids, K) 18. initial\_centroids = new\_centroids(data, K, 19. cluster, initial\_centroids) 20. *# 计算畸变程度并存储* 21. distortion = calculate\_distortion(data, cluster, 22. initial\_centroids) 23. distortions.append(distortion) 24. *# 使用肘部法找到最佳 K 值* 25. optimal\_K = find\_optimal\_K(distortions) 26. *# 返回最佳 K 对应的结果* 27. return cluster, initial\_centroids, optimal\_K,distortions 28. def calculate\_distortion(data, cluster, centroids): 29. distortion = 0 30. for i in range(len(data)): 31. distortion += np.linalg.norm(data[i] – 32. centroids[cluster[i]])\*\*2 33. return distortion 34. *#传入一组的数* 35. def find\_optimal\_K(distortions): 36. *# 使用肘部法找到最佳 K 值* 37. *# 计算每相邻两个 K 对应的畸变程度变化率* 38. *# distortions\_changes作为一个数组储存* 39. distortions\_changes = [distortions[i] - distortions[i + 1] 40. for i in range(len(distortions) - 1)] 41. *# 找到肘部，即畸变程度变化率开始减缓的位置* 42. optimal\_K\_index = 43. distortions\_changes.index(max(distortions\_changes)) 44. *# 最佳 K 值为肘部对应的 K 值加1* 45. optimal\_K = optimal\_K\_index + 2  *# 加1是因为索引从0开始，K从1开始* 46. return optimal\_K 47. *# 调用 KMeans\_train\_with\_K 函数* 48. cluster\_result, centroids\_result, optimal\_K\_value, 49. SEE = KMeans\_train\_with\_K(data, max\_K=10, max\_train=100) 50. *# 打印最优的 K 值* 51. *# 图像展示* 52. print("Best K:", optimal\_K\_value) 53. K\_values = range(1, len(SEE) + 1) 54. plt.plot(K\_values, SEE, marker='o',color='green') 55. plt.title('K') 56. plt.xlabel('Number of K') 57. plt.ylabel('SSE') 58. plt.show() 59. 、效果展示：   1、iris数据集：  在<http://archive.ics.uci.edu/>中获取iris数据集，通过观察数据集，得到iris已经分类好了三类，为'Iris-setosa','Iris-versicolor','Iris-virginica'，因此，在此处可以聚类成三类。  首先，先导入数据，并对数据进行预处理    将聚类进行比较，来查看聚类的效果。首先打印出有标签的结果    得到的结果：    打印出无标签的结果：    得到的结果：    运用kmeans算法进行聚类，设置K=3，最大迭代次数=500：    得到结果：     * + 1. Breast Cancer Wisconsin (Diagnostic)数据集   导入数据，并对数据进行降维。可以看到，wdbc数据集中有32个特征值，先将带有字母的列清洗    将数据降到2维度    使用肘部法寻找K值：  由于大量数据中的K值不易得出，因此使用手肘法。核心指标是SSE(sum of the squared errors，误差平方和)，    其中，Ci是第i个簇，p是Ci中的样本点，mi是Ci的质心（Ci中所有样本的均值），SSE是所有样本的聚类误差，代表了聚类效果的好坏。  定义KMeans\_train\_with\_K函数来获得Kmeans算法中的K值：  依旧使用距离函数来获取SEE值（用distortion代替）    通过将所有的斜率存放在distortions\_changes数组中，获取到最大值，即为下降最快的点，也就是斜率最大点，加上2后，可以得到自然数表示的K值。  最后打印图像    得到结果：    获得K值为2  将K值代入Kmeans原算法中，对data进行聚类：  无标签的情况：    聚类后的情况：    3、NPHA-doctor-visits数据集  通过数据导入，得到该数据集有714行，15个特征值，降维得到2个。并用肘部法得到K值为2    将K值代入Kmeans算法聚类得到：    4、Forty soybean cultivars from subsequent harvests数据集  先清洗数据，去掉带有字符串的项，得到319行10个特征值。进行tsne降 维后得到新的数据。    在进行聚类可得到聚类后的结果：    5、Abalone数据集  数据集中有非int和float类型的字符，将其删除：    计算K值，得到：    取K值为4，进行聚类得到：    **实验三**     * 1. **分析与数据观察**   **（以下使用jupyter notebook格式）**  打开ccf\_offline\_stage1\_test\_revised文件，观察到文件的列由“User\_id”、“Merchant\_id”、“Coupon\_id”、“Discount\_rate”、“Distance”、“Date\_received”六项组成。对数据分析可知：    对整体数据分析获得：    对数据进行统一的处理，①复制新的offline，对他进行填充一些数据。②填充Distance中的空值。③创建新的date\_received，转化时间显示模式。④找到折扣率。⑤打标。⑥添加满减列。    得到新的格式：    对数据进行可视化处理，第一步，先观察用户在哪个时间段领取了优惠券，通过教程学习，画出图像：    打开html文件，获得图像：    图：Bar\_1  该图像表示出了从2016年7月1日开始到2016年7月31日结束，优惠券领取情况。通过观察，可以得知，  多数人选择在7月3日领取优惠券，达到峰值。  添加weekday\_Receive列，用来统计用户在周几领取了优惠券    得到情况：    对周几领取优惠券进行可视化，绘制折线图：    得到图像：    由图像可知，在周日领取的人数最多，考虑到周末更多的人由空余时间购物。  判断折扣情况，对满减转化过来的折扣率进行统计：  得到折扣情况：    可以得知，大部分折扣率处于八五折，也有极不幸运的用户抽到了九九折扣，极其幸运的抽到了三折。  画出图像：    得到图像：    对商家的距离进行分析：    得到数据：    绘出图像：    得到条形图：    通过图像可以得到，大部分客户都是居住在附近的客户，也有一部分回头客（远距离客户）占到了一定比例。  查看优惠券满减和折扣占比：    得到图像：    **该数据分析与可视化完整代码：（jupyter notebook格式）**   1. #!/usr/bin/env python 2. # coding: utf-8 4. # In[2]:  7. **import** pandas as pd 8. **import** matplotlib.pyplot as plt 9. **from** pyecharts.charts **import** Bar, Line, Pie 10. **from** pyecharts **import** options as opts 11. plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  14. # In[3]:  17. data=pd.read\_csv("C:\\Users\\李昌峻\Desktop\ccf\_offline\_stage1\_test\_revised.csv") 18. offline=data.copy() 19. offline  22. # In[4]:  25. offline['Distance'].fillna(-1, downcast='infer', inplace=True) 26. offline  29. # In[5]:  32. offline['date\_received'] = pd.to\_datetime(offline['Date\_received'], format='%Y%m%d') 33. # offline['date'] = pd.to\_datetime(offline['Date'], format='%Y%m%d') 34. offline['discount\_rate'] = offline['Discount\_rate'].map(**lambda** x: float(x) **if** ':' **not** **in** str(x) **else** 35. (float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0])) 36. offline['is\_manjian'] = offline['Discount\_rate'].map(**lambda** x: 1 **if** ':' **in** str(x) **else** 0) 37. offline['Distance'].fillna(-1, downcast='infer', inplace=True) 38. offline  41. # In[6]:  44. offline.isnull().any()  47. # ### 数据大致分析 49. # In[7]:  52. sum = data.shape[0] 53. **print**(f'总数据：', sum, '条') 55. # 共有多少条优惠券的领取记录 56. received\_count = data['Date\_received'].count() 57. **print**('优惠券领取数量：', received\_count, '张') 59. # 共有多少种不同的优惠券 60. diff\_kinds = len(data['Coupon\_id'].value\_counts()) 61. **print**('优惠券种类', diff\_kinds, '种') 63. # 共有多少个用户 64. users\_num = len(data['User\_id'].value\_counts()) 65. **print**('用户数量', users\_num, '位') 67. # 共有多少个商家 68. merchant\_num = len(data['Merchant\_id'].value\_counts()) 69. **print**('商家数量', merchant\_num, '家') 71. # 最早领券时间 72. min\_received = str(int(data['Date\_received'].min())) 73. # 最晚领券时间 74. max\_received = str(int(data['Date\_received'].max())) 75. #转化为时间模式 76. min\_received=pd.to\_datetime(min\_received) 77. max\_received=pd.to\_datetime(max\_received) 79. **print**('最早领卷', min\_received) 80. **print**('最晚领卷', max\_received) 82. columns\_to\_check = ['Date\_received', 'Coupon\_id', 'Merchant\_id', 'User\_id', 'Distance', 'Discount\_rate'] 84. **for** column\_to\_check **in** columns\_to\_check: 85. missing\_values = data[column\_to\_check].isnull().sum() 86. **if** missing\_values > 0: 87. **print**(f"列 '{column\_to\_check}' 中有 {missing\_values} 个缺失值。") 88. **else**: 89. **print**(f"列 '{column\_to\_check}' 中没有缺失值。")  92. # In[18]:  95. # 每日领取优惠券的分析以及图像绘制 96. df\_1 = offline[offline['Date\_received'].notna()] 97. tmp = df\_1.groupby('Date\_received', as\_index=False)['Coupon\_id'].count() 98. bar\_1 = Bar(init\_opts=opts.InitOpts(width='1500px', height='600px')) 99. # 横纵坐标设置 100. axis\_x = list(tmp['Date\_received']) 101. axis\_y = list(tmp['Coupon\_id']) 102. # set 103. bar\_1.add\_xaxis(axis\_x) 104. bar\_1.add\_yaxis("领取数量", axis\_y) 105. bar\_1.set\_series\_opts(markline\_opts=opts.MarkLineOpts(data=[opts.MarkLineItem(type\_="max")])) 106. bar\_1.set\_global\_opts( 107. title\_opts=opts.TitleOpts(title='每天被领券的数量'),  # title 108. legend\_opts=opts.LegendOpts(is\_show=True),  # 显示ToolBox 109. xaxis\_opts=opts.AxisOpts(axislabel\_opts=opts.LabelOpts(rotate=60), interval=1),  # 旋转60度 110. ) 111. bar\_1.render("C:\\Users\\李昌峻\Desktop\绘图\课后作业五\\bar\_1.html")  114. # In[9]:  117. # 处理周几领取优惠券人数 118. # 添加领券时间为周几 119. offline['weekday\_Receive'] = offline['date\_received'].apply(**lambda** x: x.isoweekday()) 120. offline  123. # In[10]:  126. # 绘图line 127. get\_weekdays = offline[offline['weekday\_Receive'].notna()]['weekday\_Receive'].value\_counts() 128. get\_weekdays.sort\_index(inplace=True) 129. **print**(get\_weekdays) 130. line\_1 = ( 131. Line() 132. .add\_xaxis([str(x) **for** x **in** range(1, 8)]) 133. .add\_yaxis('周几领取', list(get\_weekdays)) 134. .set\_global\_opts(title\_opts={'text': '星期领取日'}) 135. .set\_series\_opts( 136. opts.LabelOpts(is\_show=True)  # 显示值大小 137. ) 138. ) 139. line\_1.render('C:\\Users\\李昌峻\\Desktop\\绘图\\课后作业五\\line\_1.html')  142. # ### 折扣情况判断 144. # In[11]:  147. # 处理打折情况 148. **print**(f'最低打折',offline['discount\_rate'].max()) 149. **print**(f'最高打折',offline['discount\_rate'].min()) 150. **print**(f'平均折扣',offline['discount\_rate'].mean()) 151. offline['discount\_rate']  154. # In[12]:  157. received = offline[['discount\_rate']] 158. received['cnt'] = 1 159. received = received.groupby('discount\_rate').agg('sum').reset\_index() 161. # 注意这里使用了 received['discount\_rate']，确保使用正确的列名 162. discount\_rate\_x = list(received['discount\_rate']) 163. discount\_rate\_y = list(received['cnt']) 165. bar\_2 = ( 166. Bar() 167. .add\_xaxis(discount\_rate\_x) 168. .add\_yaxis('领取数量', discount\_rate\_y) 169. .set\_global\_opts(title\_opts={'text': '优惠券折扣情况'}) 170. .set\_series\_opts(opts.LabelOpts(is\_show=True)) 171. ) 173. bar\_2.render('C:\\Users\\李昌峻\\Desktop\\绘图\\课后作业五\\bar\_2.html')  176. # ### 下面分析商家距离 178. # In[13]:  181. dis = offline[offline['Distance'] != -1]['Distance'].value\_counts() 182. dis.sort\_index(inplace=True) 183. dis  186. # In[14]:  189. # 绘图 190. bar\_3 = ( 191. Bar() 192. .add\_xaxis([str(x) **for** x **in** range(0, 11)]) 193. .add\_yaxis('消费距离', list(dis)) 194. .set\_global\_opts(title\_opts={'text': '卖家和买家距离条形图'}) 195. .set\_series\_opts( 196. opts.LabelOpts(is\_show=False)  # 显示值大小 197. ) 198. ) 199. bar\_3.render('C:\\Users\\李昌峻\\Desktop\\绘图\\课后作业五\\bar\_3.html')  202. # 优惠券占比图 204. # In[15]:  207. v1 = ['折扣', '满减'] 208. v2 = list(offline[offline['Date\_received'].notna()]['is\_manjian'].value\_counts()) 209. pie\_1 = ( 210. Pie() 211. .add('', [list(v) **for** v **in** zip(v1, v2)]) 212. .set\_global\_opts(title\_opts={'text': '各类优惠券数量占比饼图'}) 213. .set\_series\_opts(label\_opts=opts.LabelOpts(formatter='{b}: {c}')) 214. ) 215. pie\_1.render('C:\\Users\\李昌峻\\Desktop\\绘图\\课后作业五\\pie\_1.html')   **二、数据预处理**  同上述代码部分内容  **作业（八）**  **1、实验步骤：**  想着开始对阿里天池o2o的比赛中的数据进行提取特征，把整个任务二同时完成。分别从用户，商家和优惠券三个角度提取了一些特征，并代入到了xgboost算法中预测，但是第一次尝试效果不佳，train\_auc值达到了0.93左右，但是提交score只有0.57左右。任务一由于时间紧迫，还没有找到问题所在原因和解决方案，准备后续再来改进模型，提高auc值。  **下面是目前完成的步骤**   1. 导入数据：   数据导入的同时，对里面的np.nan格式全部转化为字符串‘null’，方便后续处理：     1. 单独对data\_train进行折扣率列添加：      1. 对数据划分：   采用时间窗口划分方法，提出train、validate、test三大数据集，并且通过中间预留15天空闲时间最为空窗期，分开了训练集、空窗期和测试集     1. 提出特征：  * 用户领券数      * 领券并消费数      * 领券未消费数      * 领券并消费数/领券数（核销）      * 领券并消费优惠券的平均折扣率      * 领券并消费的平均距离      * 在多少不同商家领取并消费优惠券      * 在多少不同商家领取优惠券      * 在多少不同商家领取并消费优惠券/在多少不同商家领取优惠券     后续也包含了对get\_Merchant\_Related\_Feature以及Coupon\_Related\_future相关特征的分析，此处完成任务一，不详细解释。  代入到xgboost模型中训练，训练中得到auc为    提交到天池后得到结果为：    成绩非常不理想，目前还未找到原因。希望在任务二的一个月时间内，能够查出原因，改进并提高auc值。  **2、完整代码展示：**   1. **import** numpy as np 2. **import** pandas as pd 3. **import** matplotlib.pyplot as plt 4. **from** pyecharts **import** options as opts 5. **import** os 6. **from** datetime **import** date 7. **import** pickle 8. **import** xgboost as xgb 9. **from** sklearn.preprocessing **import** MinMaxScaler 10. **from** sklearn.metrics **import** log\_loss, roc\_auc\_score, auc,roc\_curve 11. **from**  sklearn.model\_selection **import** train\_test\_split 12. # 使用GridSearchCV进行参数搜索 13. **from** xgboost.sklearn **import** XGBClassifier 14. **from** sklearn.model\_selection **import** GridSearchCV 15. # 绘制特征得分图 16. **import** matplotlib.pyplot as plt 17. **from** xgboost **import** plot\_importance 19. """ 20. train数据： 21. 列 'Date\_received' 中有 418751 个缺失值。 22. 列 'Coupon\_id' 中有 418751 个缺失值。 23. 列 'Merchant\_id' 中没有缺失值。 24. 列 'User\_id' 中没有缺失值。 25. 列 'Date' 中有 584858 个缺失值。 26. 列 'Distance' 中有 62986 个缺失值。 27. 列 'Discount\_rate' 中有 418751 个缺失值。 28. """ 29. **def** prepare\_data(data): 30. # 找到折扣率 31. data['discount\_rate'] = data['Discount\_rate'].map(**lambda** x: float(x) **if** ':' **not** **in** str(x) **else** 32. (float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0])) 33. # 添加满减列： 34. data['is\_manjian'] = data['Discount\_rate'].map(**lambda** x: 1 **if** ':' **in** str(x) **else** 0) 36. **return** data 38. **def** getLabel(row): # 打标 39. row = row.values 40. a = str(row[0]) 41. b = str(row[1]) 42. **if** a=='null' **or** b=='null': 43. **return** 0 44. **elif** (date(int(b[0:4]),int(b[4:6]),int(b[6:8])) - date(int(a[0:4]),int(a[4:6]),int(a[6:8]))).days <= 15: 45. **return** 1 46. **else**: 47. **return** 0 49. **def** time\_change(data): 50. data = data\_train.copy() 51. # 创建新的date\_received，date，转化时间显示模式 52. data['date\_received'] = pd.to\_datetime(data['Date\_received'], format='%Y%m%d') 53. data['date'] = pd.to\_datetime(data['Date'], format='%Y%m%d') 54. **return** data   58. **def** getReceivedUseGap(dates): 59. dates = dates.values 60. # print(dates) 61. receive,use = dates[0],dates[1] 62. **return** (date(int(use[0:4]),int(use[4:6]),int(use[6:8])) - 63. date(int(receive[0:4]),int(receive[4:6]),int(receive[6:8]))).days  66. # 用户特征分析 67. **def** get\_User\_Related\_Feature(feature): 68. """ 69. ##########提取的特征： 70. User\_receive\_from\_diff\_Merchant:用户在不同商家领取 71. User\_buy\_merchant\_count:每个用户购买商品的不同商户数量 72. User\_max\_distance：用户距离用消费券的店铺的最大值 73. User\_min\_distance：用户距离用消费券的店铺的最小值 74. User\_mean\_distance：用户距离用消费券的店铺的平均值 75. User\_median\_distance：用户距离用消费券的店铺的中位数 76. User\_buy\_use\_coupon\_count：用户使用优惠券消费次数 77. User\_buy\_count：用户总体消费次数 78. User\_received\_count：用户总共领取的消费券次数 79. User\_received\_use\_gap：用户领取了隔了几天才使用的次数 80. User\_received\_use\_max\_gap：gap天数的最大值 81. User\_received\_use\_min\_gap：gap天数的最小值 82. User\_received\_use\_mean\_gap：gap天数的平均值 83. User\_browser\_count：用户总数 84. Discount\_rate\_mean：用户领券并消费部分的平均数 85. User\_no\_buy\_use\_coupon\_count：用户领券但是没有消费的数量 86. ##概率： 87. User\_buy\_use\_coupon\_goumai\_rate：用户消费中使用优惠券率 88. user\_buy\_use\_coupon\_hexiao\_rate：用户领券中使用优惠券概率（核销率）  91. :param feature: 92. :return: 93. """ 95. t = feature['User\_id'].copy() 96. t.drop\_duplicates(inplace=True) 98. # 特征：用户在不同商家领取 99. t12=feature[feature['Date\_received']!='null'][['User\_id','Merchant\_id']].copy() 100. t12['User\_receive\_from\_diff\_Merchant']=1 101. t12.groupby(['User\_id']).agg('sum').reset\_index() 103. # 特征：用户在不同商家领券消费类数 104. t1 = feature[(feature['Date']!='null')&(feature['Date\_received']!='null')][['User\_id','Merchant\_id']].copy() 105. # t1.drop\_duplicates(inplace=True) 106. t1['User\_buy\_from\_diff\_Merchant'] = 1 107. t1 = t1.groupby('User\_id').agg('sum').reset\_index() 108. # t1.rename(columns={'Merchant\_id':'User\_buy\_from\_diff\_Merchant'},inplace=True)  111. t2 = feature[(feature['Date']!='null') & (feature['Coupon\_id']!='null')][['User\_id','Distance']].copy() 112. t2.replace('null',-1,inplace=True) 113. t2['Distance'] = t2['Distance'].astype(float) 114. t2.replace(-1,np.nan,inplace=True) 116. # 特征：用户距离已用消费券消费店铺的最大、最小、平均、中位距离 117. t2\_1 = t2.groupby('User\_id').agg('max').reset\_index() 118. t2\_1.rename(columns={'Distance':'User\_max\_distance'},inplace=True) 119. t2\_2 = t2.groupby('User\_id').agg('min').reset\_index() 120. t2\_2.rename(columns={'Distance':'User\_min\_distance'},inplace=True) 121. t2\_3 = t2.groupby('User\_id').agg('mean').reset\_index() 122. t2\_3.rename(columns={'Distance':'User\_mean\_distance'},inplace=True) 123. t2\_4 = t2.groupby('User\_id').agg('median').reset\_index() 124. t2\_4.rename(columns={'Distance':'User\_median\_distance'},inplace=True) 126. #特征：用户使用优惠券并消费次数 127. t3 = feature[(feature['Coupon\_id']!='null') & (feature['Date']!='null')][['User\_id']].copy() 128. t3['User\_buy\_use\_coupon\_count'] = 1 129. t3 = t3.groupby('User\_id').agg('sum').reset\_index() 131. # 特征：用户消费次数 132. t4 = feature[(feature['Date']!='null')][['User\_id']].copy() 133. t4['User\_buy\_count'] = 1 134. t4 = t4.groupby('User\_id').agg('sum').reset\_index() 136. # 特征：用户领券优惠券数目（总数） 137. t5 = feature[(feature['Coupon\_id'] != 'null')][['User\_id']].copy() 138. t5['User\_received\_count'] = 1 139. t5 = t5.groupby('User\_id').agg('sum').reset\_index() 141. # 特征：用户领券并使用消费券间隔天数 142. t6 = feature[(feature['Coupon\_id'] != 'null') & (feature['Date'] != 'null')][['User\_id', 'Date\_received', 'Date']].copy() 143. t6['User\_received\_use\_gap'] = t6[['Date\_received', 'Date']].apply(getReceivedUseGap, axis=1) 144. t6 = t6[['User\_id', 'User\_received\_use\_gap']] 146. # 特征：用户领券并使用优惠券的最大/最小/平均间隔天数 147. t7 = t6.copy() 148. t7\_1 = t7.groupby('User\_id').agg('max').reset\_index() 149. t7\_1.rename(columns={'User\_received\_use\_gap':'User\_received\_use\_max\_gap'},inplace=True) 150. t7\_2 = t7.groupby('User\_id').agg('min').reset\_index() 151. t7\_2.rename(columns={'User\_received\_use\_gap':'User\_received\_use\_min\_gap'},inplace=True) 152. t7\_3 = t7.groupby('User\_id').agg('mean').reset\_index() 153. t7\_3.rename(columns={'User\_received\_use\_gap':'User\_received\_use\_mean\_gap'},inplace=True) 155. # 特征：用户总数 156. t8 = feature[['User\_id']].copy() 157. t8['User\_browser\_count'] = 1 158. t8 = t8.groupby('User\_id').agg('sum').reset\_index()   162. #特征：用户领取优惠券但没有消费次数 163. t10 = feature[(feature['Coupon\_id']!='null') & (feature['Date']=='null')][['User\_id']].copy() 164. t10['User\_no\_buy\_use\_coupon\_count'] = 1 165. t10 = t10.groupby('User\_id').agg('sum').reset\_index() 167. #特征：用户领取并消费的平均折扣率： 168. #找到折扣率 169. t11=feature[(feature['Coupon\_id']!='null')&(feature['discount\_rate']!='null')][['User\_id','discount\_rate']] 171. t11=t11.groupby(['User\_id']).agg('mean').reset\_index() 172. t11=t11.rename(columns={'discount\_rate':'discount\_rate\_mean'},inplace=True)   176. userFeature = pd.merge(t,t1,on='User\_id',how='left') 177. # userFeature = pd.merge(userFeature, t1, on='User\_id', how='left') 178. userFeature = pd.merge(userFeature,t2\_1,on='User\_id',how='left') 179. userFeature = pd.merge(userFeature,t2\_2,on='User\_id',how='left') 180. userFeature = pd.merge(userFeature,t2\_3,on='User\_id',how='left') 181. userFeature = pd.merge(userFeature,t2\_4,on='User\_id',how='left') 182. userFeature = pd.merge(userFeature,t3,on='User\_id',how='left') 183. userFeature = pd.merge(userFeature,t4,on='User\_id',how='left') 184. userFeature = pd.merge(userFeature,t5,on='User\_id',how='left') 185. userFeature = pd.merge(userFeature,t6,on='User\_id',how='left') 186. userFeature = pd.merge(userFeature,t7\_1,on='User\_id',how='left') 187. userFeature = pd.merge(userFeature,t7\_2,on='User\_id',how='left') 188. userFeature = pd.merge(userFeature,t7\_3,on='User\_id',how='left') 189. userFeature = pd.merge(userFeature,t8,on='User\_id',how='left') 190. userFeature = pd.merge(userFeature, t10, on='User\_id', how='left') 191. userFeature = pd.merge(userFeature, t12, on='User\_id', how='left')  194. # 特征：客户使用优惠券率：用户消费总量中使用优惠券占比 195. userFeature['User\_buy\_use\_coupon\_goumai\_rate'] = (userFeature['User\_buy\_use\_coupon\_count'] 196. /userFeature['User\_buy\_count']) 197. # 特征：优惠券的核销率：用户领券并消费的数目/用户领券数目 198. userFeature['user\_buy\_use\_coupon\_hexiao\_rate'] = (userFeature['User\_buy\_use\_coupon\_count'] 199. /userFeature['User\_received\_count']) 200. # 特征：用户在不同商家领取消费/在商家领取种数：定义为商家选取率： 201. userFeature['seller\_been\_chosen']=(userFeature['User\_buy\_from\_diff\_Merchant'] 202. /userFeature['User\_receive\_from\_diff\_Merchant'])  205. # 对于次数或者数目或占比，将Nan转换为0 206. userFeature['User\_buy\_from\_diff\_Merchant'].replace(np.nan,0,inplace=True) 207. userFeature['User\_buy\_use\_coupon\_count'].replace(np.nan,0,inplace=True) 208. userFeature['User\_buy\_count'].replace(np.nan,0,inplace=True) 209. userFeature['User\_received\_count'].replace(np.nan,0,inplace=True) 210. userFeature['User\_buy\_use\_coupon\_goumai\_rate'].replace(np.nan,0,inplace=True) 211. userFeature['user\_buy\_use\_coupon\_hexiao\_rate'].replace(np.nan, 0, inplace=True) 212. userFeature['seller\_been\_chosen'].replace(np.nan,0,inplace=True) 213. userFeature['User\_no\_buy\_use\_coupon\_count'].replace(np.nan,0,inplace=True) 215. **return** userFeature 217. # 商家特征分析 218. **def** get\_Merchant\_Related\_Feature(feature): 220. t = feature['Merchant\_id'].copy() 221. t.drop\_duplicates(inplace=True) 223. # 特征：商家卖出数目 224. t1 = feature[(feature['Date']!='null')][['Merchant\_id']].copy() 225. t1['Merchant\_sale\_count'] = 1 226. t1 = t1.groupby('Merchant\_id').agg('sum').reset\_index() 228. # 特征：商家核销数目 229. t2 = feature[(feature['Coupon\_id']!='null') & (feature['Date']!='null')][['Merchant\_id']].copy() 230. t2['Merchant\_sale\_use\_coupon\_count'] = 1 231. t2 = t2.groupby('Merchant\_id').agg('sum').reset\_index() 233. # 特征：商家优惠券的总数量 234. t3 = feature[(feature['Coupon\_id']!='null')][['Merchant\_id']].copy() 235. t3['Merchant\_give\_count'] = 1 236. t3 = t3.groupby('Merchant\_id').agg('sum').reset\_index() 238. t4 = feature[(feature['Coupon\_id']!='null') & (feature['Date']!='null')][['Merchant\_id','Distance']].copy() 239. t4['Distance'].replace('null',-1,inplace=True) 240. t4['Distance'] = t4['Distance'].astype(float) 241. t4['Distance'].replace(-1,np.nan,inplace=True) 242. # 特征：商家已核销优惠券中距离的最小\最大\平均\中值 243. t4\_1 = t4.groupby('Merchant\_id').agg('max').reset\_index() 244. t4\_1.rename(columns={'Distance':'Merchant\_max\_distance'},inplace=True) 245. t4\_2 = t4.groupby('Merchant\_id').agg('min').reset\_index() 246. t4\_2.rename(columns={'Distance':'Merchant\_min\_distance'},inplace=True) 247. t4\_3 = t4.groupby('Merchant\_id').agg('mean').reset\_index() 248. t4\_3.rename(columns={'Distance':'Merchant\_mean\_distance'},inplace=True) 250. merchantFeature = pd.merge(t,t1,on='Merchant\_id',how='left') 251. merchantFeature = pd.merge(merchantFeature,t2,on='Merchant\_id',how='left') 252. merchantFeature = pd.merge(merchantFeature,t3,on='Merchant\_id',how='left') 253. merchantFeature = pd.merge(merchantFeature,t4\_1,on='Merchant\_id',how='left') 254. merchantFeature = pd.merge(merchantFeature,t4\_2,on='Merchant\_id',how='left') 255. merchantFeature = pd.merge(merchantFeature,t4\_3,on='Merchant\_id',how='left') 257. # 特征：商家卖出总量中优惠券的核销比 258. merchantFeature['Merchant\_sale\_use\_coupon\_rate'] = (merchantFeature['Merchant\_sale\_use\_coupon\_count'] 259. /merchantFeature['Merchant\_sale\_count']) 260. # 特征：商家发放总量中优惠券的核销比 261. merchantFeature['Merhcant\_give\_coupon\_use\_rate'] = (merchantFeature['Merchant\_sale\_use\_coupon\_count'] 262. /merchantFeature['Merchant\_give\_count']) 264. # 次数项目和占比类型数据，Nan用0替代(之所以最后转化，是防止上两个特征提取时出现分母为零溢出)（另外，上两个特征值的计算，只要分子分母一个为pd.nan结果就为nd.nan） 265. merchantFeature['Merchant\_sale\_use\_coupon\_count'].replace(np.nan,0,inplace=True) 266. merchantFeature['Merchant\_sale\_count'].replace(np.nan,0,inplace=True) 267. merchantFeature['Merchant\_give\_count'].replace(np.nan,0,inplace=True) 268. merchantFeature['Merchant\_sale\_use\_coupon\_rate'].replace(np.nan,0,inplace=True) 269. merchantFeature['Merhcant\_give\_coupon\_use\_rate'].replace(np.nan,0,inplace=True)  272. **return** merchantFeature 274. # 优惠券特征分析 275. **def** Coupon\_Related\_future(dataset): 276. """ 277. Coupon\_give\_weekday:消费券发放的星期几 278. Coupon\_give\_monthday：消费券发放的月份几号数目 279. Coupon\_discount\_type：是否满减 280. Coupon\_discount\_man：满多少触发 281. Coupon\_discount\_jian：减多少触发 282. Coupon\_discount\_rate：打折率 283. Coupon\_count：数目 284. :param dataset: 285. :return: 286. """ 288. t = dataset.copy() 289. # 这里dataset无重复值，不用drop\_duplicates() 291. # 特征：消费券发放的周或月份 292. #20160101  对时间数字进行切片处理 293. t['Coupon\_give\_weekday'] = t['Date\_received'].astype(str).apply( 294. **lambda** x: date(int(x[0:4]), int(x[4:6]), int(x[6:8])).weekday() + 1) 295. t['Coupon\_give\_monthday'] = t['Date\_received'].astype(str).apply(**lambda** x: int(x[6:8])) 297. t['Discount\_rate'] = t['Discount\_rate'].astype(str) 298. # 特征：消费券是否是满减类型   ：表示满减多少 299. t['Coupon\_discount\_type'] = t['Discount\_rate'].apply(**lambda** s: 1 **if** ':' **in** s **else** 0) 300. # 特征：消费券满减的满 301. t['Coupon\_discount\_man'] = t['Discount\_rate'].apply(**lambda** s: int(s.split(':')[0]) **if** ':' **in** s **else** 0) 302. # 特征：消费券减 303. t['Coupon\_discount\_jian'] = t['Discount\_rate'].apply(**lambda** s: int(s.split(':')[1]) **if** ':' **in** s **else** 0) 304. # 特征：优惠券打折率（在写一次） 305. t['Coupon\_discount\_rate'] = t['Discount\_rate'].map(**lambda** x: float(x) **if** ':' **not** **in** str(x) **else** 306. (float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0])) 308. # 特征：每种优惠券的数目 309. t1 = dataset[['Coupon\_id']].copy() 310. t1['Coupon\_count'] = 1 311. t1 = t1.groupby('Coupon\_id').agg('sum').reset\_index() 313. couponFeature = pd.merge(t, t1, on='Coupon\_id', how='left')  316. **return** couponFeature 318. **def** isWeekend(day): 319. **if** day>=1 **and** day<=5: 320. **return** 0 321. **else**: 322. **return** 1 324. **def** featureProcess(dataset,feature,processFlag): 325. user = get\_User\_Related\_Feature(feature) 326. merchant = get\_Merchant\_Related\_Feature(feature) 327. coupon = Coupon\_Related\_future(dataset)  330. allFeature = pd.merge(coupon,user,on='User\_id',how='left') 331. allFeature = pd.merge(allFeature,merchant,on='Merchant\_id',how='left')  334. allFeature['Coupon\_give\_weekday\_is\_weekend'] = allFeature['Coupon\_give\_weekday'].apply(isWeekend) 335. weekday\_dummies = pd.get\_dummies(allFeature['Coupon\_give\_weekday']) 336. weekday\_dummies.columns = ['Coupon\_give\_weekday\_' + str(i) **for** i **in** range(1,weekday\_dummies.shape[1]+1)] 337. allFeature = pd.concat([allFeature,weekday\_dummies],axis=1) 338. allFeature.drop('Coupon\_give\_weekday',axis=1,inplace=True) 340. **if** processFlag: 341. allFeature['Label'] = allFeature[['Date\_received','Date']].apply(getLabel,axis=1) 342. allFeature.drop(['User\_id','Date\_received','Coupon\_id','Merchant\_id','Discount\_rate','Date'],axis=1,inplace=True) 343. **else**: 344. # 'User\_id','Date\_received','Coupon\_id'字段需要在提交文档中,先留下 345. allFeature.drop(['Merchant\_id','Discount\_rate'],axis=1,inplace=True) 346. allFeature.replace('null',np.nan,inplace=True) 348. **return** allFeature    353. **if** \_\_name\_\_ == '\_\_main\_\_': 355. data\_train = pd.read\_csv('D:\\Data\\opodata\\tabel3\\ccf\_offline\_stage1\_train.csv', 356. header=0, keep\_default\_na=False) 357. data\_test = pd.read\_csv('D:\\Data\\opodata\\tabel1\\ccf\_offline\_stage1\_test\_revised.csv', 358. header=0, keep\_default\_na=False) 359. # 找到折扣率（data\_train) 360. data\_train['Discount\_rate'] = data\_train['Discount\_rate'].replace('null', np.nan) 361. data\_train['discount\_rate'] = data\_train['Discount\_rate'].map(**lambda** x: float(x) **if** ':' **not** **in** str(x) **else** 362. (float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0])) 363. data\_train['discount\_rate'] = data\_train['discount\_rate'].replace(np.nan,'null') 365. data\_test['discount\_rate'] = data\_test['Discount\_rate'].map(**lambda** x: float(x) **if** ':' **not** **in** str(x) **else** 366. (float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0])) 367. **print**(data\_test.head(5)) 368. **print**(data\_train.head(5)) 370. Path = r'D:\Data\opodata' 372. # 划分区间 373. # 训练集特征 374. # 提取特征 375. train\_history\_field = data\_train[(data\_train.Date\_received >= '20160302') 376. & (data\_train.Date\_received <= '20160501')] 377. train\_middle\_field = data\_train[(data\_train.Date\_received >= '20160501') 378. & (data\_train.Date\_received <= '20160516')] 379. train\_label\_field = data\_train[(data\_train.Date\_received >= '20160516') 380. & (data\_train.Date\_received <= '20160616')] 382. # 验证集特征 383. validate\_history\_field = data\_train[(data\_train.Date\_received >= '20160116') 384. & (data\_train.Date\_received <= '20160316')] 385. validate\_middle\_field = data\_train[(data\_train.Date\_received >= '20160316') 386. & (data\_train.Date\_received <= '20160331')] 387. validate\_label\_field = data\_train[(data\_train.Date\_received >= '20160331') 388. & (data\_train.Date\_received <= '20160501')] 390. # 测试集特征 391. test\_history\_field = data\_train[(data\_train.Date\_received >= '20160417') 392. & (data\_train.Date\_received <= '20160616')] 393. test\_middle\_field = data\_train[(data\_train.Date\_received >= '20160616') 394. & (data\_train.Date\_received <= '20160701')] 395. test\_label\_field = data\_train[(data\_train.Date\_received >= '20160701') 396. & (data\_train.Date\_received <= '20160801')] 398. # 验证test文件 399. test\_label\_field = data\_test.copy()  # test作为测试集 400. test\_label\_field['Date\_received'] = test\_label\_field['Date\_received'].astype(str)  403. df1 = featureProcess(train\_label\_field, train\_history\_field, True)  # train 404. df1.to\_csv(Path + r'\df1.csv') 405. **print**('df1 write over') 407. df2 = featureProcess(validate\_label\_field, validate\_history\_field, True)  #  validate 408. df2.to\_csv(Path + r'\df2.csv') 409. **print**('df2 write over') 411. df3 = featureProcess(test\_label\_field, test\_history\_field, False)  # test 412. df3.to\_csv(Path + r'\df3.csv') 413. **print**('df3 write over') 415. ##########################       训练 416. ######### 新数据 417. Path = r'D:\Data\opodata' 419. train = pd.read\_csv(Path + r'\df1.csv', index\_col=0) 420. validate = pd.read\_csv(Path + r'\df2.csv', index\_col=0) 421. test = pd.read\_csv(Path + r'\df3.csv', index\_col=0) 422. # 输出保留三列 423. **print**(train.columns) 424. test\_preds = test[['User\_id', 'Coupon\_id', 'Date\_received']].copy() 425. test\_x = test.drop(['User\_id', 'Coupon\_id', 'Date\_received'], axis=1) 427. dataset\_12 = pd.concat([train, validate], axis=0) 428. dataset\_12\_y = dataset\_12.Label 429. dataset\_12\_x = dataset\_12.drop(['Label'], axis=1) 431. dataTrain = xgb.DMatrix(dataset\_12\_x, label=dataset\_12\_y) 432. dataTest = xgb.DMatrix(test\_x) 433. **print**('---data prepare over---') 435. params = {'booster': 'gbtree', 436. 'objective': 'binary:logistic', 437. 'eval\_metric': 'auc', 438. 'gamma': 0, 439. 'min\_child\_weight': 1.1, 440. 'max\_depth': 5, 441. 'lambda': 10, 442. 'subsample': 0.9, 443. 'colsample\_bytree': 0.7, 444. 'colsample\_bylevel': 0.7, 445. 'eta': 0.05, 446. 'tree\_method': 'exact', 447. 'seed': 0, 448. } 450. watchlist = [(dataTrain, 'train')] 451. model = xgb.train(params, dataTrain, num\_boost\_round=1000, evals=watchlist) 452. # 然后进行预测 454. **print**('start predict') 455. test\_preds1 = test\_preds 456. test\_preds1['Label'] = model.predict(dataTest) 457. **print**(type(test\_preds1.Label)) 458. test\_preds1['Label'] = MinMaxScaler(copy=True, feature\_range=(0, 1)).fit\_transform( 459. test\_preds1['Label'].values.reshape(-1, 1)) 460. test\_preds1.to\_csv(Path + r'\sample\_submission.csv', index=None, header=True) 461. **print**('write over') | | | | |
| **四、学习心得** | | | | |
| 作为一名大一新生，完成整个项目不太顺利。报名过后，才第一次开始接触到了python，先前只是学习了C语言。在2024年1月16日期末考试结束过后，开始正式学习，通过大量查阅CSDN、知乎上对于python的资料，在B站上浏览学习相关视频，逐渐对python语法熟悉并能够运用。接着了解到了python中比较常用的Numpy库以及Pandas库的使用，并在实验一中体现出来一些简单的数据分析和绘图。  后来就接触到了机器学习中的第一个回归算法，Kmeans算法。通过大量资料查询，了解到了Kmeans的底层逻辑，并通过代码实现了出来。由于这是第一次完成一个python语言的项目，因此耗费了我许多的时间，并且代码也不够非常完美。但是却让我更加了解到python这门语言，并且后续通过数据集进行Kmeans算法的实现也让我对数据分析有了一定的了解  O2O优惠券预测项目是新人对数据分析入门较好的项目选择，同时在完成该任务时，我学习到了简单的数据分析、通过pyecharts绘图、数据打标、提取特征值、时间窗口划分法等一系列操作。并调用了xgboost算法对train和validate数据集进行测试，预测test中7月过后用户消费的概率。但是发现，即使我提取到了较多的特征，似乎在score上并没有太多的提升。于是在接下来的任务二中，我将继续学习机器学习、数据挖掘等科目，希望能够取得好的score分数，并且更加熟悉数据分析流程。 | | | | |
| **五、教师评阅意见** | | | | |
|  | | | | |