**餐饮智能推荐服务项目分析报告**

何梓维

**加载包：**

#载入模块  
import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np

**任务1.1：**

**完成思路：**

将数据文件放进项目路径中，使用pandas库的read\_csv函数即可读取csv文件，然后读取查看数据形状是否正确。

**操作步骤：**

#任务1.1：载入数据  
data = pd.read\_csv('meal\_order\_detail.csv')  
infodata = pd.read\_csv('meal\_order\_info.csv')  
print('原detail表数据形状：',data.shape)  
data\_wash = data

**任务结果：**

原detail表数据形状： (10037, 19)

**结果分析：**

数据读入正确。

**任务1.2：**

**完成思路：**

由于detail表中的菜品名称夹杂有换行号和空格号，需要将这些符号清除。首先使用datafram数据类型的replace函数，将换行号均换成空格，regex=True代表正则替换。然后str.strip函数去除所有的空格号。

**操作步骤：**

#任务1.2：去除菜品名称中的空格和回车  
data\_wash = data\_wash.replace('\n','', regex=True)  
data\_wash.loc[:,'dishes\_name']=data\_wash.loc[:,'dishes\_name'].str.strip()

**结果分析：**

data\_wash中已不含有换行号和空格号，数据清洗成功。

**任务1.3：**

**完成思路：**

1.首先因为白饭不属于菜品，所以将数据中的白饭订单统统除去。通过索引检索（index属性）的方式将所有dishes\_name为“白饭/小碗”和“白饭/大碗”的行删除（drop函数）。

2.然后统计各个菜品的销售数量，就是将同一菜品的订单的counts相加，具体操作是：先通过去除dishes\_name的重复值（drop\_duplicates函数）获得菜单列表，然后遍历菜单，将相同菜名的count相加，最后将结果放到一个以菜名为键、以销量为值的菜品销量字典中。

3.通过菜品销量字典获得菜品热销度字典，公式是：

是菜品i的热销度，是菜品i的销量，是所有菜品中的最小销量，是所有菜品中的最大销量。得到菜品热销度字典后以热销度为值进行从大到小的排序（sorted函数）。

**操作步骤：**

#任务1.3：各个菜品的热销度评分  
#在已清洗数据中除去白饭  
data\_wash = data\_wash.drop(index=(data\_wash.loc[(data\_wash['dishes\_name']=='白饭/小碗')].index))  
data\_wash = data\_wash.drop(index=(data\_wash.loc[(data\_wash['dishes\_name']=='白饭/大碗')].index))  
#统计各个菜品的销售数量  
dish\_name = data\_wash['dishes\_name']  
dish\_name = list(dish\_name.drop\_duplicates())  
dish\_num = {}  
for i in dish\_name:  
 dish\_num[i] = 0  
for i in dish\_name:  
 for j,h in zip(data\_wash['dishes\_name'],data\_wash['counts']):  
 if i==j:  
 dish\_num[i] = dish\_num[i]+h  
dish\_num\_sorted = sorted(dish\_num.items(), key=lambda x: x[1],reverse=True)  
print('各个菜品的销售数量分别为：',dish\_num\_sorted)  
#创建菜品热销度字典  
min\_sale = dish\_num\_sorted[-1][1] #最小销量  
max\_sale = dish\_num\_sorted[0][1] #最大销量  
dish\_sale = {}  
for i in dish\_name:  
 dish\_sale[i] = (dish\_num[i]-min\_sale)/(max\_sale-min\_sale)  
dish\_sale\_sorted = sorted(dish\_sale.items(), key=lambda x: x[1],reverse=True)  
print('菜品热销度分别为：',dish\_sale\_sorted)

**任务结果：**

各个菜品的销售数量分别为： [('凉拌菠菜', 272), ('谷稻小庄', 243), ('麻辣小龙虾', 219), ('五色糯米饭(七色)', 195), ('蛋挞', 192), ('芝士烩波士顿龙虾', 190), ('辣炒鱿鱼', 189), ('香酥两吃大虾', 179), ('焖猪手', 174), ('水煮鱼', 162), ('蒙古烤羊腿', 158), ('西瓜胡萝卜沙拉', 136), ('自制猪肉脯', 136), ('美妙绝伦之白莲花', 135), ('北冰洋汽水', 134), ('孜然羊排', 132), ('剁椒鱼头', 129), ('香烤牛排', 128), ('爆炒猪肝', 124), ('意文柠檬汁', 121), ('凉拌蒜蓉西兰花', 118), ('清蒸海鱼', 117), ('芹菜炒腰花', 112), ('豌豆薯仔猪骨汤', 112), ('番茄炖牛腩', 111), ('炝炒大白菜', 108), ('姜葱炒花蟹', 107), ('油焖麻辣虾', 106), ('白胡椒胡萝卜羊肉汤', 105), ('冰糖红豆薏米粥', 104), ('蒜香辣花甲', 102), ('皮蛋瘦肉粥', 101), ('香菇鹌鹑蛋', 100), ('糖醋蕃茄溜青花', 100), ('爆炒双丝', 98), ('桂圆肉红豆八宝饭', 95), ('番茄有机花菜', 94), ('独家薄荷鲜虾牛肉卷', 92), ('山药养生粥', 90), ('青岛啤酒罐装', 86), ('香辣腐乳炒虾', 86), ('拌土豆丝', 85), ('大蒜苋菜', 85), ('农夫山泉NFC果汁100%橙汁', 84), ('番茄甘蓝', 80), ('小米南瓜粥', 80), ('清蒸蝶鱼', 80), ('培根紫菜卷', 78), ('金玉良缘', 76), ('清爽拌凉面', 76), ('酸辣藕丁', 74), ('白斩鸡', 73), ('倒立蒸梭子蟹', 73), ('凉拌萝卜丝', 71), ('香辣爆羊肝', 70), ('哈尔滨啤酒罐装', 69), ('蓝带啤酒罐装', 69), ('葱油凉拌藕片', 67), ('啤酒鸭', 66), ('培根花菜', 66), ('红酒炖羊肉', 66), ('照烧鸡腿', 64), ('番茄炖秋葵', 64), ('干锅田鸡', 63), ('爆炒鸡翅', 63), ('黄花菜炒木耳', 62), ('南瓜枸杞小饼干', 62), ('桂圆枸杞鸽子汤', 61), ('肉丁茄子', 61), ('盘蟹蒸蛋', 61), ('露露无糖杏仁露', 60), ('百里香奶油烤红酒牛肉', 59), ('清炒菊花菜', 58), ('黑米恋上葡萄', 58), ('芝麻烤紫菜', 57), ('木须豌豆', 56), ('紫薯面包卷', 56), ('避风塘炒蟹', 55), ('黄尾袋鼠西拉子红葡萄酒', 53), ('张裕葡萄酒张裕赤霞珠干红', 53), ('牛尾汤', 53), ('葱姜炒蟹', 52), ('五香酱驴肉', 52), ('香菇鸡肉粥', 51), ('一品香酥藕', 51), ('路易拉菲红酒干红', 50), ('蒜香包', 50), ('辣炒田螺', 49), ('红酒土豆烧鸭腿', 49), ('打卤面', 49), ('蒜蓉生蚝', 48), ('大理石奶油蛋糕', 47), ('南瓜芝士包', 46), ('葡萄奶酥(超酥香的饼干)', 44), ('泡椒凤爪', 43), ('番茄蛋汤', 42), ('党参黄芪炖牛尾', 41), ('青炒扁豆', 41), ('纸杯蛋糕', 41), ('五彩藕苗', 41), ('法国波尔多AOC干红葡萄酒原', 41), ('花蛤蒸蛋', 40), ('鱼香肉丝拌面', 40), ('53度茅台', 39), ('辣炒海带丝', 39), ('不加一滴油的酸奶蛋糕', 38), ('杭椒鸡珍', 37), ('百威啤酒罐装', 36), ('辣炒花蛤', 36), ('超人气广式肠粉', 34), ('老式面包', 33), ('核桃葡萄干土司', 32), ('小炒羊腰', 32), ('牛肉鸡蛋肠粉', 30), ('重庆特色油烧兔', 29), ('五彩豆', 29), ('黄油曲奇饼干', 28), ('海带结豆腐汤', 28), ('酸辣汤面', 27), ('42度海之蓝', 25), ('38度剑南春', 24), ('牛奶卷', 23), ('爆炒鳝碌', 23), ('玉竹南北杏鸭腿汤', 23), ('三色凉拌手撕兔', 22), ('广式虾仁肠粉', 22), ('鲜美鳝鱼', 22), ('52度泸州老窖', 22), ('50度古井贡酒', 21), ('冬瓜炒苦瓜', 20), ('咖啡奶香面包', 19), ('特醇嘉士伯啤酒罐装', 19), ('土豆西红柿汤面', 19), ('快炒黄鳝', 19), ('凉拌海带豆芽紫菜', 18), ('芹黄鳝丝', 18), ('长城窖酿解百纳红酒干红葡萄酒', 18), ('宝宝版牛肉肠粉', 17), ('冰镇花螺', 17), ('鸡蛋、肉末肠粉', 14), ('三丝鳝鱼', 10), ('百里香奶油烤紅酒牛肉', 5), ('铁板牛肉', 3)]

菜品热销度分别为： [('凉拌菠菜', 1.0), ('谷稻小庄', 0.8921933085501859), ('麻辣小龙虾', 0.8029739776951673), ('五色糯米饭(七色)', 0.7137546468401487), ('蛋挞', 0.7026022304832714), ('芝士烩波士顿龙虾', 0.6951672862453532), ('辣炒鱿鱼', 0.6914498141263941), ('香酥两吃大虾', 0.654275092936803), ('焖猪手', 0.6356877323420075), ('水煮鱼', 0.5910780669144982), ('蒙古烤羊腿', 0.5762081784386617), ('西瓜胡萝卜沙拉', 0.4944237918215613), ('自制猪肉脯', 0.4944237918215613), ('美妙绝伦之白莲花', 0.49070631970260226), ('北冰洋汽水', 0.48698884758364314), ('孜然羊排', 0.4795539033457249), ('剁椒鱼头', 0.4684014869888476), ('香烤牛排', 0.4646840148698885), ('爆炒猪肝', 0.44981412639405205), ('意文柠檬汁', 0.43866171003717475), ('凉拌蒜蓉西兰花', 0.4275092936802974), ('清蒸海鱼', 0.42379182156133827), ('芹菜炒腰花', 0.4052044609665427), ('豌豆薯仔猪骨汤', 0.4052044609665427), ('番茄炖牛腩', 0.40148698884758366), ('炝炒大白菜', 0.3903345724907063), ('姜葱炒花蟹', 0.38661710037174724), ('油焖麻辣虾', 0.3828996282527881), ('白胡椒胡萝卜羊肉汤', 0.379182156133829), ('冰糖红豆薏米粥', 0.3754646840148699), ('蒜香辣花甲', 0.3680297397769517), ('皮蛋瘦肉粥', 0.3643122676579926), ('香菇鹌鹑蛋', 0.36059479553903345), ('糖醋蕃茄溜青花', 0.36059479553903345), ('爆炒双丝', 0.35315985130111527), ('桂圆肉红豆八宝饭', 0.3420074349442379), ('番茄有机花菜', 0.3382899628252788), ('独家薄荷鲜虾牛肉卷', 0.3308550185873606), ('山药养生粥', 0.32342007434944237), ('青岛啤酒罐装', 0.30855018587360594), ('香辣腐乳炒虾', 0.30855018587360594), ('拌土豆丝', 0.3048327137546468), ('大蒜苋菜', 0.3048327137546468), ('农夫山泉NFC果汁100%橙汁', 0.30111524163568776), ('番茄甘蓝', 0.2862453531598513), ('小米南瓜粥', 0.2862453531598513), ('清蒸蝶鱼', 0.2862453531598513), ('培根紫菜卷', 0.2788104089219331), ('金玉良缘', 0.27137546468401486), ('清爽拌凉面', 0.27137546468401486), ('酸辣藕丁', 0.26394052044609667), ('白斩鸡', 0.26022304832713755), ('倒立蒸梭子蟹', 0.26022304832713755), ('凉拌萝卜丝', 0.2527881040892193), ('香辣爆羊肝', 0.24907063197026022), ('哈尔滨啤酒罐装', 0.24535315985130113), ('蓝带啤酒罐装', 0.24535315985130113), ('葱油凉拌藕片', 0.2379182156133829), ('啤酒鸭', 0.2342007434944238), ('培根花菜', 0.2342007434944238), ('红酒炖羊肉', 0.2342007434944238), ('照烧鸡腿', 0.22676579925650558), ('番茄炖秋葵', 0.22676579925650558), ('干锅田鸡', 0.22304832713754646), ('爆炒鸡翅', 0.22304832713754646), ('黄花菜炒木耳', 0.21933085501858737), ('南瓜枸杞小饼干', 0.21933085501858737), ('桂圆枸杞鸽子汤', 0.21561338289962825), ('肉丁茄子', 0.21561338289962825), ('盘蟹蒸蛋', 0.21561338289962825), ('露露无糖杏仁露', 0.21189591078066913), ('百里香奶油烤红酒牛肉', 0.20817843866171004), ('清炒菊花菜', 0.20446096654275092), ('黑米恋上葡萄', 0.20446096654275092), ('芝麻烤紫菜', 0.20074349442379183), ('木须豌豆', 0.1970260223048327), ('紫薯面包卷', 0.1970260223048327), ('避风塘炒蟹', 0.19330855018587362), ('黄尾袋鼠西拉子红葡萄酒', 0.18587360594795538), ('张裕葡萄酒张裕赤霞珠干红', 0.18587360594795538), ('牛尾汤', 0.18587360594795538), ('葱姜炒蟹', 0.1821561338289963), ('五香酱驴肉', 0.1821561338289963), ('香菇鸡肉粥', 0.17843866171003717), ('一品香酥藕', 0.17843866171003717), ('路易拉菲红酒干红', 0.17472118959107807), ('蒜香包', 0.17472118959107807), ('辣炒田螺', 0.17100371747211895), ('红酒土豆烧鸭腿', 0.17100371747211895), ('打卤面', 0.17100371747211895), ('蒜蓉生蚝', 0.16728624535315986), ('大理石奶油蛋糕', 0.16356877323420074), ('南瓜芝士包', 0.15985130111524162), ('葡萄奶酥(超酥香的饼干)', 0.1524163568773234), ('泡椒凤爪', 0.14869888475836432), ('番茄蛋汤', 0.1449814126394052), ('党参黄芪炖牛尾', 0.1412639405204461), ('青炒扁豆', 0.1412639405204461), ('纸杯蛋糕', 0.1412639405204461), ('五彩藕苗', 0.1412639405204461), ('法国波尔多AOC干红葡萄酒原', 0.1412639405204461), ('花蛤蒸蛋', 0.137546468401487), ('鱼香肉丝拌面', 0.137546468401487), ('53度茅台', 0.13382899628252787), ('辣炒海带丝', 0.13382899628252787), ('不加一滴油的酸奶蛋糕', 0.13011152416356878), ('杭椒鸡珍', 0.12639405204460966), ('百威啤酒罐装', 0.12267657992565056), ('辣炒花蛤', 0.12267657992565056), ('超人气广式肠粉', 0.11524163568773234), ('老式面包', 0.11152416356877323), ('核桃葡萄干土司', 0.10780669144981413), ('小炒羊腰', 0.10780669144981413), ('牛肉鸡蛋肠粉', 0.10037174721189591), ('重庆特色油烧兔', 0.09665427509293681), ('五彩豆', 0.09665427509293681), ('黄油曲奇饼干', 0.09293680297397769), ('海带结豆腐汤', 0.09293680297397769), ('酸辣汤面', 0.08921933085501858), ('42度海之蓝', 0.08178438661710037), ('38度剑南春', 0.07806691449814127), ('牛奶卷', 0.07434944237918216), ('爆炒鳝碌', 0.07434944237918216), ('玉竹南北杏鸭腿汤', 0.07434944237918216), ('三色凉拌手撕兔', 0.07063197026022305), ('广式虾仁肠粉', 0.07063197026022305), ('鲜美鳝鱼', 0.07063197026022305), ('52度泸州老窖', 0.07063197026022305), ('50度古井贡酒', 0.06691449814126393), ('冬瓜炒苦瓜', 0.06319702602230483), ('咖啡奶香面包', 0.05947955390334572), ('特醇嘉士伯啤酒罐装', 0.05947955390334572), ('土豆西红柿汤面', 0.05947955390334572), ('快炒黄鳝', 0.05947955390334572), ('凉拌海带豆芽紫菜', 0.055762081784386616), ('芹黄鳝丝', 0.055762081784386616), ('长城窖酿解百纳红酒干红葡萄酒', 0.055762081784386616), ('宝宝版牛肉肠粉', 0.05204460966542751), ('冰镇花螺', 0.05204460966542751), ('鸡蛋、肉末肠粉', 0.040892193308550186), ('三丝鳝鱼', 0.026022304832713755), ('百里香奶油烤紅酒牛肉', 0.007434944237918215), ('铁板牛肉', 0.0)]

**结果分析：**

1.白饭没有在后面的结果中出现，数据清洗成功。

2.销量字典的顺序和热销度字典一致，字典创建成功。

3.在所有菜品中，凉拌菠菜销量达到272，最多；铁板牛肉最小，只有3.

**任务1.4：**

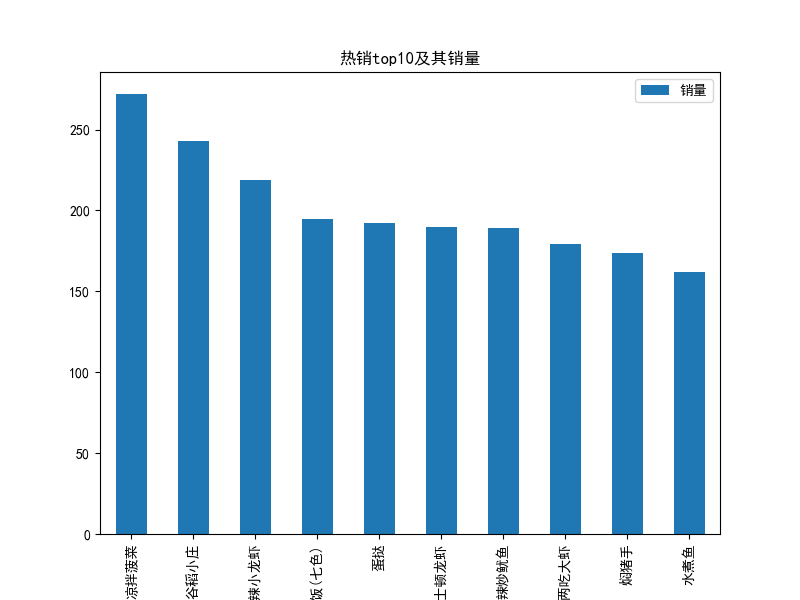
**完成思路：**

由于已经创建了销量字典，所以只需取销量前10的菜品绘图即可。因为plt绘图默认不显示中文，需要用rcParams手动设置字体。为方便plot函数调用，创建10行2列的数据框，kind=’bar’意为条形图，grid=’False’不标记网格。最后记得plt.show().

**操作步骤：**

#任务1.4：绘制条形图展示热销top10  
#找到热销top10及其销量  
dish\_salenum\_top10 = dish\_num\_sorted[0:10]  
#绘制条形图  
plt.rcParams['figure.figsize'] = (8.0, 6.0)  
plt.rcParams['font.sans-serif']=['SimHei'] #这行代码用于显示中文，'SimHei'就是黑体  
dish\_salenum\_top10 = pd.DataFrame(dish\_salenum\_top10,columns=['菜品','销量'])  
dish\_salenum\_top10.plot(y='销量',x='菜品',kind='bar',grid=False,title='热销top10及其销量')  
plt.show()

**任务结果：**



**结果分析：**

前十销量分别为：('凉拌菠菜', 272), ('谷稻小庄', 243), ('麻辣小龙虾', 219), ('五色糯米饭(七色)', 195), ('蛋挞', 192), ('芝士烩波士顿龙虾', 190), ('辣炒鱿鱼', 189), ('香酥两吃大虾', 179), ('焖猪手', 174), ('水煮鱼', 162)。前三名间均差距25左右；而四五六七名销量差距不大，都在190左右。

**任务2.1：**

**完成思路：**

在info表中只需统计0，1，2的个数，再除以总数即可。

**操作步骤：**

#任务2.1：统计各种订单状态所占比例  
order\_status\_counts = infodata['order\_status'].value\_counts()  
print('各种订单状态所占比例（1为正常）：',order\_status\_counts/len(infodata))

**任务结果：**

各种订单状态所占比例（1为正常）：

1 0.987302

0 0.009524

2 0.003175

Name: order\_status, dtype: float64

**结果分析：**

可以看到正常状态的订单高达0.987302，即便将不正常的订单去掉也不影响模型构建，故后面只留下状态1的订单。

**任务2.2：**

**完成思路：**

1.采用索引和drop函数除去info表里状态不正常的订单。

2.因为有些订单在detail表里有，而info表里没有，这些订单是非法的，需要除去。首先在detail表和info表里统一订单id的名称。然后基于info表创建一个订单id列表，如果detail表里的订单不存在于该订单id列表中，说明其非法。for遍历detail表里的所有订单id，n作为两表一致的计数器，n=0时说明该订单id不存在于订单列表中，需要删去。

**操作步骤：**

#任务2.2：除去无效的订单，只保留正常的订单  
infodata\_wash = infodata.drop(index=(infodata.loc[(infodata['order\_status']==0)].index))  
infodata\_wash = infodata\_wash.drop(index=(infodata\_wash.loc[(infodata\_wash['order\_status']==2)].index))  
#数据清洗：将info表中有而detail表中没有，或info表中没有而detail表中有的订单除去  
infodata\_wash.rename(columns={'info\_id': 'order\_id'}, inplace=True) #将两表的订单id名称统一  
order\_id = infodata\_wash['order\_id']  
order\_id = list(order\_id.drop\_duplicates())  
for i in data\_wash['order\_id']:  
 n = 0  
 for j in order\_id:  
 if i==j:  
 n=n+1  
 if n==0:  
 data\_wash = data\_wash.drop(index=(data\_wash.loc[(data\_wash['order\_id'] == i)].index))  
print('清洗后的detail表形状：',data\_wash.shape)

**任务结果：**

清洗后的detail表形状： (9436, 19)

**结果分析：**

可以看到，比对第一读入时的数据形状(10037,19)，601行被删去了。数据清洗成功。

**任务2.3：**

**完成思路：**

由于菜品推荐模型只需要用到菜名（dishes\_name）和用户ID（emp\_id）两个特征，故将其它特征都删掉。

**操作步骤：**

#任务2.3：保留主要特征emp\_id（用户id）和dishes\_name（菜品名称）  
data\_wash\_columns = list(data\_wash.columns)  
for i in data\_wash\_columns:  
 if i!='emp\_id' and i!='dishes\_name':  
 data\_wash.drop(i,axis=1,inplace=True)  
print('进一步清洗后（保留主要特征后）的detail表形状：',data\_wash.shape)

**任务结果：**

进一步清洗后（保留主要特征后）的detail表形状： (9436, 2)

**结果分析：**

比对上一步的数据形状，只剩下了两个特征，数据预处理成功。

**任务3.1：**

**完成思路：**

1.按用户id来划分训练集（70%）和测试集（30%）。由于划分是随机的，应该设置随机种子，这里设置为6。首先创建用户id列表（emp\_id），然后在用户id列表中遍历。对每个用户id创建一个0到10的随机整数，如果该整数小于3（即0，1，2），则将该用户id分到测试集，否则分到训练集。这样就确保了训练集和测试集之间是7：3.

2.根据上一步里已划分出来的用户id训练测试集，在洗后的detail数据中删去用户id测试集的行，得到训练集数据；在洗后的detail数据中删去用户id训练集的行，得到测试集数据。

**操作步骤：**

#任务3.1：按用户id来划分训练集（70%）和测试集（30%）  
#划分训练集用户和测试集用户  
emp\_id = data\_wash['emp\_id']  
emp\_id = list(emp\_id.drop\_duplicates())  
test\_emp\_id = []  
train\_emp\_id = []  
np.random.seed(6) #设置随机种子  
for i in emp\_id:  
 if np.random.randint(0,10)<3:  
 test\_emp\_id.append(i)  
 else:  
 train\_emp\_id.append(i)  
print('测试集用户数：',len(test\_emp\_id))  
print('训练集用户数：',len(train\_emp\_id))  
#划分出dataframe格式的训练集和测试集  
test\_data = data\_wash  
train\_data = data\_wash  
for i in test\_emp\_id:  
 train\_data = train\_data.drop(index=(train\_data.loc[(train\_data['emp\_id']==i)].index))  
for i in train\_emp\_id:  
 test\_data = test\_data.drop(index=(test\_data.loc[(test\_data['emp\_id']==i)].index))  
print('测试集形状',test\_data.shape)  
print('训练集形状',train\_data.shape)

**任务结果：**

测试集用户数： 152

训练集用户数： 317

测试集形状 (2873, 2)

训练集形状 (6563, 2)

**结果分析：**

1.152/(152+317)=0.3241，用户id测试集占比32.41%，划分成功。

2.2873/(2873+6563)=0.3045，菜品数据测试集占比30.45%，划分成功。

**任务3.2：**

**完成思路：**

创建一个训练集用户-菜品二维矩阵，第一层按菜品名，排序与dish\_name列表相同；第二层按用户id，排序与train\_emp\_id相同。其中，元素为1代表对应行的菜品被对应列的用户id购买了，否则为0。首先把该矩阵的形状构建出来（numpy的mat），所有元素均为0（numpy的zeros）。然后两层for遍历菜品和用户id，如果该用户有购买该菜品，则在训练集数据中可以找到该菜品和用户id（empty判断返回结果是否为空，即能不能找到），从而用户-菜品二维矩阵的对应位置元素改为1.

**操作步骤：**

#任务3.2：构建训练集用户-菜品二维矩阵  
m = range(len(dish\_name))  
n = range(len(train\_emp\_id))  
train\_matrix = np.mat(np.zeros((len(dish\_name),len(train\_emp\_id)))) #第一层按菜品名，排序与dish\_name列表相同；第二层按用户id，排序与train\_emp\_id相同。  
for i in m:  
 for j in n:  
 if train\_data[(train\_data.loc[:,'dishes\_name']==dish\_name[i]) & (train\_data.loc[:,'emp\_id']==train\_emp\_id[j])].empty==False:  
 train\_matrix[i,j]=1 #该循环消耗时间很长  
print('训练集用户-菜品二维矩阵为:',train\_matrix)

**任务结果：**

训练集用户-菜品二维矩阵为:

[[1. 0. 0. ... 0. 0. 0.]

[1. 1. 1. ... 0. 1. 0.]

[1. 0. 0. ... 0. 0. 0.]

...

[0. 0. 0. ... 0. 0. 0.]

[0. 0. 0. ... 0. 0. 0.]

[0. 0. 0. ... 0. 0. 0.]]

**结果分析：**

0-1矩阵符合预期，用户-菜品二维矩阵创建成功。

**任务4.1：**

**完成思路：**

Jaccard相似度的计算公式：

采用基于物品的协同推荐过滤算法，A菜品和B菜品的Jaccard相似度为：A菜品和B菜品的相同购买用户数量除以两菜品购买用户的并的数量。首先创建一个两层矩阵，都是按照dish\_name列表的数量。然后两层for遍历菜品名称，第一层为菜品A，第二层为菜品B，cross为交数量，union为并数量。接着在两层遍历里再遍历一遍训练集用户id，在训练集用户-菜品二维矩阵中找到该用户有无购买菜品A、菜品B的记录。如果都有，交数量cross+1；如果其中之一有，并数量union+1.训练集用户id遍历结束后，用cross除以union得到菜品A和菜品B的Jaccard相似度。

**操作步骤：**

#任务4.1：采用Jaccard相似度方法计算相似度矩阵  
similar\_matrix = np.mat(np.zeros((len(dish\_name),len(dish\_name)))) #菜品与菜品之间的Jaccard相似度作为值，菜品排序与dish\_name相同。  
for i in m:  
 for h in m:  
 cross = 0  
 union = 0  
 for j in n:  
 if train\_matrix[i,j]==1 and train\_matrix[h,j]==1:  
 cross=cross+1  
 if train\_matrix[i,j]==1 or train\_matrix[h,j]==1:  
 union=union+1  
 similar\_matrix[i,h]=cross/union  
print('菜品相似度矩阵为：',similar\_matrix)

**任务结果：**

菜品相似度矩阵为：

[[1. 0.17610063 0.18181818 ... 0.08139535 0.04938272 0.04819277]

[0.17610063 1. 0.14492754 ... 0.06349206 0.05 0.05785124]

[0.18181818 0.14492754 1. ... 0.08474576 0.01818182 0.03571429]

...

[0.08139535 0.06349206 0.08474576 ... 1. 0. 0.03030303]

[0.04938272 0.05 0.01818182 ... 0. 1. 0.04 ]

[0.04819277 0.05785124 0.03571429 ... 0.03030303 0.04 1. ]]

**结果分析：**

该矩阵为对称矩阵，且对角线为1，符合两菜品之间的相似度规则，菜品相似度矩阵创建成功。

**任务4.2：**

**完成思路：**

用户i对于菜品的兴趣分公式为：

其中，为该用户已购买的菜品，为需要计算兴趣分的菜品，为已购买菜品和需计算兴趣分的菜品之间的相似度，对所有这些相似度求和即得到用户i对于菜品的兴趣分。

以测试集用户为目标客户，创建关于测试集用户的菜品推荐列表。首先确定该矩阵的形状：第一层是按测试集用户id，第二层是按菜品名dish\_name，元素均设为0。然后两层for分别遍历测试集用户id和菜品名，找到该用户购买过的所有菜品，将这些已购买菜品和当前菜品的相似度求和，即得到该用户对该菜品的兴趣分。

**操作步骤：**

#任务4.2：针对目标客户生成推荐列表，这里以测试集用户为目标客户。若A已点且只点了b、c，则A用户对b菜品的兴趣分为(1\*b与b的相似度+1\*b与c的相似度)  
test\_emp\_interest\_matrix = np.mat(np.zeros((len(test\_emp\_id),len(dish\_name)))) #创建测试集用户的菜品推荐矩阵， #第一层按用户id，排序与test\_emp\_id列表相同；第二层按菜品名，排序与dish\_name相同。  
for j in range(len(test\_emp\_id)):  
 test\_emp\_dishdata = test\_data[(test\_data.loc[:, 'emp\_id'] == test\_emp\_id[j])] # 将当前客户点的所有菜品检索出  
 test\_emp\_dishdata = list(test\_emp\_dishdata['dishes\_name'])  
 for i in m:  
 interest\_score = 0  
 for h in test\_emp\_dishdata:  
 index = dish\_name.index(h)  
 interest\_score = interest\_score + 1\*similar\_matrix[i,index]  
 test\_emp\_interest\_matrix[j,i] = interest\_score  
print('测试集用户的推荐列表：',test\_emp\_interest\_matrix)

**任务结果：**

测试集用户的推荐列表：

[[12.35804763 12.84392666 9.10623604 ... 5.11224601 4.01207298

3.02927555]

[ 3.67612899 4.00004023 3.14365623 ... 1.56605767 1.42819671

1.0199065 ]

[ 1.00250575 1.06152296 0.95621121 ... 0.42515803 0.30427273

0.24031168]

...

[ 2.33226703 3.18961865 1.89928803 ... 2.05938896 0.96417552

0.69248465]

[ 1.28530152 2.17073351 1.02732695 ... 0.48705192 0.40881925

0.27689217]

[ 2.342446 2.06771819 1.67276702 ... 1.02163948 0.86107147

0.49829298]]

**结果分析：**

推荐列表中的兴趣分为多个相似度相加，如果要得到具体菜品的推荐列表，只需取兴趣分排前n的菜品的索引，通过索引在菜名表里找到具体菜名，再放到一个以该用户为键的字典里。由于推荐菜品的种类数量可多可少，我们希望推荐的刚好是已购买的，便采用用户已购买菜品的数量作为推荐菜品的种类数量（当然其它整数都可以）。下一步将得到用户已购买菜品的字典。

**任务5.1：**

**完成思路：**

遍历测试集用户id，在测试集数据中找出该用户购买过的所有菜品。从而创建一个已点菜品字典，以用户id为键，以该用户已购买菜品列表为值的字典。

操作步骤：

#任务5.1：构建测试集客户的已点菜品字典  
test\_emp\_data = {}  
for j in range(len(test\_emp\_id)):  
 test\_emp\_dishdata = test\_data[(test\_data.loc[:, 'emp\_id'] == test\_emp\_id[j])] # 将当前客户点的所有菜品检索出  
 test\_emp\_dishdata = list(test\_emp\_dishdata['dishes\_name'])  
 test\_emp\_data[test\_emp\_id[j]] = test\_emp\_dishdata  
print('测试集客户的已点菜品字典',test\_emp\_data)

**任务结果：**

测试集客户的已点菜品字典 {1097: ['芝士烩波士顿龙虾', '清蒸海鱼', '百里香奶油烤紅酒牛肉', '辣炒鱿鱼', '爆炒猪肝', '啤酒鸭', '干锅田鸡', '番茄蛋汤', '黄花菜炒木耳', '西瓜胡萝卜沙拉', '凉拌萝卜丝', '38度剑南春', '蓝带啤酒罐装', '老式面包', '一品香酥藕', '牛奶卷', '葡萄奶酥(超酥香的饼干)', '水煮鱼', '大蒜苋菜', '芝麻烤紫菜', '路易拉菲红酒干红', '白胡椒胡萝卜羊肉汤', '凉拌菠菜', '百里香奶油烤红酒牛肉', '牛肉鸡蛋肠粉', '蛋挞', '自制猪肉脯', '香酥两吃大虾', '哈尔滨啤酒罐装', '桂圆肉红豆八宝饭', '葱油凉拌藕片', '山药养生粥', '老式面包', '谷稻小庄', '蒙古烤羊腿', '香酥两吃大虾', '培根花菜', '杭椒鸡珍', '路易拉菲红酒干红', '白胡椒胡萝卜羊肉汤', '凉拌菠菜', '芝麻烤紫菜', '照烧鸡腿', '酸辣藕丁', '孜然羊排', '蛋挞', '大蒜苋菜', '红酒炖羊肉', '油焖麻辣虾', '焖猪手', '辣炒鱿鱼', '自制猪肉脯', '剁椒鱼头', '凉拌菠菜', '芹菜炒腰花', '爆炒猪肝', '蒙古烤羊腿', '干锅田鸡', '谷稻小庄', '孜然羊排', '酸辣藕丁', '南瓜枸杞小饼干', '照烧鸡腿', '泡椒凤爪', '辣炒花蛤', '凉拌蒜蓉西兰花', '糖醋蕃茄溜青花', '爆炒鸡翅', '蛋挞', '清爽拌凉面'], 990: ['芝士烩波士顿龙虾', '凉拌菠菜', '核桃葡萄干土司', '倒立蒸梭子蟹', '番茄甘蓝', '葱姜炒蟹', '白胡椒胡萝卜羊肉汤', '干锅田鸡', '木须豌豆', '露露无糖杏仁露', '剁椒鱼头', '独家薄荷鲜虾牛肉卷', '爆炒猪肝', '啤酒鸭', '肉丁茄子', '番茄有机花菜', '桂圆肉红豆八宝饭', '葡萄奶酥(超酥香的饼干)', '水煮鱼', '美妙绝伦之白莲花', '蒜香辣花甲', '小炒羊腰', '肉丁茄子', '拌土豆丝'],

... ...

, 1558: ['麻辣小龙虾', '剁椒鱼头', '独家薄荷鲜虾牛肉卷', '番茄炖牛腩', '番茄蛋汤', '凉拌蒜蓉西兰花', '五彩豆', '路易拉菲红酒干红', '肉丁茄子', '葡萄奶酥(超酥香的饼干)', '一品香酥藕', '五色糯米饭(七色)', '桂圆肉红豆八宝饭', '培根紫菜卷', '辣炒海带丝'], 1244: ['麻辣小龙虾', '辣炒花蛤', '凉拌菠菜', '大蒜苋菜', '糖醋蕃茄溜青花', '蛋挞', '清爽拌凉面'], 1380: ['香菇鹌鹑蛋', '葱油凉拌藕片', '牛尾汤', '南瓜枸杞小饼干', '不加一滴油的酸奶蛋糕', '五色糯米饭(七色)', '花蛤蒸蛋', '黑米恋上葡萄', '香烤牛排', '小米南瓜粥', '清蒸蝶鱼', '盘蟹蒸蛋', '山药养生粥']}

**结果分析：**

比对原来的detail表，已点菜品字典的格式和内容符合预期，任务完成良好。

**任务5.2：**

**完成思路：**

1.对每一个测试集用户都计算推荐准确度，i用户的准确度为：

其中，T为用户实际购买的菜品集合，S为推荐该用户购买的菜品集合，S元素的个数与T元素个数相等（这样设置是因为：如果购买的菜品均为推荐的菜品，那么推荐准确度便为1；只要有一个推荐菜品没有被购买，准确度便不能达到1），num()为返回集合元素个数的函数。

2.整体模型的推荐准确度为：，它是所有测试集用户的准确度的均值。

**操作步骤：**

#任务5.2：构建评价指标，分析模型推荐效果  
test\_emp\_accuracy = {} #构建测试集用户的推荐准确度字典  
for j in range(len(test\_emp\_id)):  
 interest\_len = len(test\_emp\_data[test\_emp\_id[j]]) #设置系统应推荐给当前客户的菜品种类与该客户的实际点菜种类相等。  
 true\_len = 0  
 a = test\_emp\_interest\_matrix[j,:].tolist()[0] #当前用户的菜品推荐列表  
 b = sorted(a,reverse=True)  
 for i in b[0:interest\_len]: #推荐列表中兴趣分排名靠前的菜品（只取排前面的系统应推荐量的名次）即为系统推荐菜品  
 if dish\_name[a.index(i)] in test\_emp\_data[test\_emp\_id[j]]:  
 true\_len = true\_len + 1 #当前用户的实际已点菜品中存在推荐的菜品，每有一个，准确数+1  
 test\_emp\_accuracy[test\_emp\_id[j]] = true\_len/interest\_len  
print('测试集用户的推荐准确度字典',test\_emp\_accuracy)  
all\_accuracy = 0 #计算模型整体准确度，为所有测试集用户推荐准确度的均值  
for j in range(len(test\_emp\_id)):  
 all\_accuracy = all\_accuracy + test\_emp\_accuracy[test\_emp\_id[j]]  
all\_accuracy = all\_accuracy/len(test\_emp\_id)  
print('模型总体准确率为：',all\_accuracy)

**任务结果：**

测试集用户的推荐准确度字典 {1097: 0.5285714285714286, 990: 0.5416666666666666, 1463: 1.0, 1486: 0.8571428571428571, 1141: 0.53125, 992: 0.6666666666666666, 1084: 0.5689655172413793, 996: 0.6428571428571429, 1250: 1.0, 1073: 0.45161290322580644, 1002: 0.5384615384615384, 991: 0.5, 1318: 1.0, 1231: 1.0, 993: 0.5405405405405406, 1095: 0.5294117647058824, 1458: 0.6363636363636364, 1468: 1.0, 1303: 1.0, 1251: 1.0, 1029: 0.5652173913043478, 1249: 1.0, 1230: 1.0, 1520: 0.6666666666666666, 1610: 1.0, 1189: 0.5185185185185185, 1206: 0.8333333333333334, 1148: 0.6666666666666666, 1068: 0.7333333333333333, 1523: 1.0, 1210: 1.0, 1108: 0.5742574257425742, 1456: 1.0, 1025: 0.5882352941176471, 1209: 1.0, 1147: 0.5116279069767442, 1579: 0.5833333333333334, 1377: 0.5714285714285714, 1534: 0.8571428571428571, 1609: 0.7692307692307693, 1473: 0.6923076923076923, 1104: 0.5238095238095238, 1355: 1.0, 1136: 0.4722222222222222, 1111: 0.6216216216216216, 1195: 0.6111111111111112, 1024: 0.6111111111111112, 1454: 0.8181818181818182, 1310: 0.7142857142857143, 1019: 0.5, 1596: 1.0, 1157: 0.5714285714285714, 1158: 0.56, 1028: 0.631578947368421, 1551: 0.5238095238095238, 1159: 0.48717948717948717, 1238: 0.8888888888888888, 1122: 0.5657894736842105, 1203: 1.0, 1418: 0.6666666666666666, 1132: 0.44680851063829785, 1127: 0.5555555555555556, 1151: 0.4406779661016949, 1275: 0.47619047619047616, 1016: 0.47540983606557374, 1027: 0.44, 1194: 0.6428571428571429, 1406: 0.6666666666666666, 1487: 0.65, 1178: 0.5833333333333334, 1217: 0.8181818181818182, 1165: 0.5348837209302325, 1571: 1.0, 1451: 0.875, 1133: 0.5217391304347826, 1123: 0.5384615384615384, 1188: 0.7692307692307693, 1078: 0.5277777777777778, 1184: 0.5882352941176471, 1022: 0.6428571428571429, 1291: 0.8888888888888888, 1322: 0.6, 1212: 0.6, 1171: 0.5555555555555556, 1502: 1.0, 1083: 0.5384615384615384, 1259: 0.7142857142857143, 1274: 0.5833333333333334, 1342: 0.8, 1130: 0.6063829787234043, 1082: 0.5454545454545454, 1197: 0.7, 1185: 0.5833333333333334, 1306: 0.9, 1240: 0.6470588235294118, 1515: 0.875, 1193: 1.0, 1204: 0.6875, 1407: 0.6666666666666666, 1362: 0.5833333333333334, 1026: 0.6111111111111112, 1575: 1.0, 1555: 1.0, 1431: 0.5882352941176471, 1435: 0.8571428571428571, 1601: 0.875, 1448: 0.8333333333333334, 1200: 1.0, 1205: 0.6086956521739131, 1598: 0.9230769230769231, 1517: 0.6666666666666666, 1576: 0.7777777777777778, 1483: 0.5294117647058824, 1453: 1.0, 1215: 0.5714285714285714, 1006: 0.8888888888888888, 1295: 0.6, 1241: 0.6153846153846154, 1574: 1.0, 1211: 0.47368421052631576, 1237: 0.5625, 1152: 0.8, 1573: 0.5789473684210527, 1196: 0.9090909090909091, 1239: 0.55, 1415: 0.5333333333333333, 1272: 0.6363636363636364, 1464: 0.7777777777777778, 1560: 1.0, 1321: 1.0, 1603: 0.55, 1535: 0.8, 1542: 1.0, 1263: 0.5714285714285714, 1363: 0.6, 1012: 1.0, 1447: 0.6363636363636364, 1351: 0.6666666666666666, 1588: 1.0, 1182: 0.8333333333333334, 1529: 0.5789473684210527, 1477: 0.8181818181818182, 1566: 0.5714285714285714, 1444: 0.7857142857142857, 1599: 1.0, 1553: 0.8888888888888888, 1516: 1.0, 1300: 0.6923076923076923, 1388: 1.0, 1558: 0.5333333333333333, 1244: 0.8571428571428571, 1380: 0.8461538461538461}

模型总体准确率为： 0.721706302997154

**结果分析：**

测试集用户的预测准确率极大部分超过0.5，模型总体准确率达到0.72，表现效果良好。

**全部代码：**

#载入模块  
import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np  
  
#任务1.1：载入数据  
data = pd.read\_csv('meal\_order\_detail.csv')  
infodata = pd.read\_csv('meal\_order\_info.csv')  
print('原detail表数据形状：',data.shape)  
data\_wash = data  
  
#任务1.2：去除菜品名称中的空格和回车  
data\_wash = data\_wash.replace('\n','', regex=True)  
data\_wash.loc[:,'dishes\_name']=data\_wash.loc[:,'dishes\_name'].str.strip()  
  
#任务1.3：各个菜品的热销度评分  
#在已清洗数据中除去白饭  
data\_wash = data\_wash.drop(index=(data\_wash.loc[(data\_wash['dishes\_name']=='白饭/小碗')].index))  
data\_wash = data\_wash.drop(index=(data\_wash.loc[(data\_wash['dishes\_name']=='白饭/大碗')].index))  
#统计各个菜品的销售数量  
dish\_name = data\_wash['dishes\_name']  
dish\_name = list(dish\_name.drop\_duplicates())  
dish\_num = {}  
for i in dish\_name:  
 dish\_num[i] = 0  
for i in dish\_name:  
 for j,h in zip(data\_wash['dishes\_name'],data\_wash['counts']):  
 if i==j:  
 dish\_num[i] = dish\_num[i]+h  
dish\_num\_sorted = sorted(dish\_num.items(), key=lambda x: x[1],reverse=True)  
print('各个菜品的销售数量分别为：',dish\_num\_sorted)  
#创建菜品热销度字典  
min\_sale = dish\_num\_sorted[0][1] #最小销量  
max\_sale = dish\_num\_sorted[-1][1] #最大销量  
dish\_sale = {}  
for i in dish\_name:  
 dish\_sale[i] = (dish\_num[i]-min\_sale)/(max\_sale-min\_sale)  
dish\_sale\_sorted = sorted(dish\_sale.items(), key=lambda x: x[1],reverse=True)  
print('菜品热销度分别为：',dish\_sale\_sorted)  
  
#任务1.4：绘制条形图展示热销top10  
#找到热销top10及其销量  
dish\_salenum\_top10 = dish\_num\_sorted[0:10]  
#绘制条形图  
plt.rcParams['font.sans-serif']=['SimHei'] #这行代码用于显示中文，'SimHei'就是黑体  
dish\_salenum\_top10 = pd.DataFrame(dish\_salenum\_top10,columns=['菜品','销量'])  
dish\_salenum\_top10.plot(y='销量',x='菜品',kind='bar',grid=False,title='热销top10及其销量')  
plt.show()  
  
#任务2.1：统计各种订单状态所占比例  
order\_status\_counts = infodata['order\_status'].value\_counts()  
print('各种订单状态所占比例（1为正常）：',order\_status\_counts/len(infodata))  
  
#任务2.2：除去无效的订单，只保留正常的订单  
infodata\_wash = infodata.drop(index=(infodata.loc[(infodata['order\_status']==0)].index))  
infodata\_wash = infodata\_wash.drop(index=(infodata\_wash.loc[(infodata\_wash['order\_status']==2)].index))  
#数据清洗：将info表中有而detail表中没有，或info表中没有而detail表中有的订单除去  
infodata\_wash.rename(columns={'info\_id': 'order\_id'}, inplace=True) #将两表的订单id名称统一  
order\_id = infodata\_wash['order\_id']  
order\_id = list(order\_id.drop\_duplicates())  
for i in data\_wash['order\_id']:  
 n = 0  
 for j in order\_id:  
 if i==j:  
 n=n+1  
 if n==0:  
 data\_wash = data\_wash.drop(index=(data\_wash.loc[(data\_wash['order\_id'] == i)].index))  
print('清洗后的detail表形状：',data\_wash.shape)  
  
#任务2.3：保留主要特征emp\_id（用户id）和dishes\_name（菜品名称）  
data\_wash\_columns = list(data\_wash.columns)  
for i in data\_wash\_columns:  
 if i!='emp\_id' and i!='dishes\_name':  
 data\_wash.drop(i,axis=1,inplace=True)  
print('进一步清洗后（保留主要特征后）的detail表形状：',data\_wash.shape)  
  
#任务3.1：按用户id来划分训练集（70%）和测试集（30%）  
#划分训练集用户和测试集用户  
emp\_id = data\_wash['emp\_id']  
emp\_id = list(emp\_id.drop\_duplicates())  
test\_emp\_id = []  
train\_emp\_id = []  
np.random.seed(6) #设置随机种子  
for i in emp\_id:  
 if np.random.randint(0,10)<3:  
 test\_emp\_id.append(i)  
 else:  
 train\_emp\_id.append(i)  
print('测试集用户数：',len(test\_emp\_id))  
print('训练集用户数：',len(train\_emp\_id))  
#划分出dataframe格式的训练集和测试集  
test\_data = data\_wash  
train\_data = data\_wash  
for i in test\_emp\_id:  
 train\_data = train\_data.drop(index=(train\_data.loc[(train\_data['emp\_id']==i)].index))  
for i in train\_emp\_id:  
 test\_data = test\_data.drop(index=(test\_data.loc[(test\_data['emp\_id']==i)].index))  
print('测试集形状',test\_data.shape)  
print('训练集形状',train\_data.shape)  
  
#任务3.2：构建训练集用户-菜品二维矩阵  
m = range(len(dish\_name))  
n = range(len(train\_emp\_id))  
train\_matrix = np.mat(np.zeros((len(dish\_name),len(train\_emp\_id)))) #第一层按菜品名，排序与dish\_name列表相同；第二层按用户id，排序与train\_emp\_id相同。  
for i in m:  
 for j in n:  
 if train\_data[(train\_data.loc[:,'dishes\_name']==dish\_name[i]) & (train\_data.loc[:,'emp\_id']==train\_emp\_id[j])].empty==False:  
 train\_matrix[i,j]=1 #该循环消耗时间很长  
print('训练集用户-菜品二维矩阵为:',train\_matrix)  
  
#模型构建：采用基于物品的协同过滤算法  
#任务4.1：采用Jaccard相似度方法计算相似度矩阵  
similar\_matrix = np.mat(np.zeros((len(dish\_name),len(dish\_name)))) #菜品与菜品之间的Jaccard相似度作为值，菜品排序与dish\_name相同。  
for i in m:  
 for h in m:  
 cross = 0  
 union = 0  
 for j in n:  
 if train\_matrix[i,j]==1 and train\_matrix[h,j]==1:  
 cross=cross+1  
 if train\_matrix[i,j]==1 or train\_matrix[h,j]==1:  
 union=union+1  
 similar\_matrix[i,h]=cross/union  
print('菜品相似度矩阵为：',similar\_matrix)  
  
#任务4.2：针对目标客户生成推荐列表，这里以测试集用户为目标客户。若A已点且只点了b、c，则A用户对b菜品的兴趣分为(1\*b与b的相似度+1\*b与c的相似度)  
test\_emp\_interest\_matrix = np.mat(np.zeros((len(test\_emp\_id),len(dish\_name)))) #创建测试集用户的菜品推荐矩阵， #第一层按用户id，排序与test\_emp\_id列表相同；第二层按菜品名，排序与dish\_name相同。  
for j in range(len(test\_emp\_id)):  
 test\_emp\_dishdata = test\_data[(test\_data.loc[:, 'emp\_id'] == test\_emp\_id[j])] # 将当前客户点的所有菜品检索出  
 test\_emp\_dishdata = list(test\_emp\_dishdata['dishes\_name'])  
 for i in m:  
 interest\_score = 0  
 for h in test\_emp\_dishdata:  
 index = dish\_name.index(h)  
 interest\_score = interest\_score + 1\*similar\_matrix[i,index]  
 test\_emp\_interest\_matrix[j,i] = interest\_score  
print('测试集用户的推荐列表：',test\_emp\_interest\_matrix)  
  
#任务5.1：构建测试集客户的已点菜品字典  
test\_emp\_data = {}  
for j in range(len(test\_emp\_id)):  
 test\_emp\_dishdata = test\_data[(test\_data.loc[:, 'emp\_id'] == test\_emp\_id[j])] # 将当前客户点的所有菜品检索出  
 test\_emp\_dishdata = list(test\_emp\_dishdata['dishes\_name'])  
 test\_emp\_data[test\_emp\_id[j]] = test\_emp\_dishdata  
print('测试集客户的已点菜品字典',test\_emp\_data)  
  
#任务5.2：构建评价指标，分析模型推荐效果  
test\_emp\_accuracy = {} #构建测试集用户的推荐准确度字典  
for j in range(len(test\_emp\_id)):  
 interest\_len = len(test\_emp\_data[test\_emp\_id[j]]) #设置系统应推荐给当前客户的菜品种类与该客户的实际点菜种类相等。  
 true\_len = 0  
 a = test\_emp\_interest\_matrix[j,:].tolist()[0] #当前用户的菜品推荐列表  
 b = sorted(a,reverse=True)  
 for i in b[0:interest\_len]: #推荐列表中兴趣分排名靠前的菜品（只取排前面的系统应推荐量的名次）即为系统推荐菜品  
 if dish\_name[a.index(i)] in test\_emp\_data[test\_emp\_id[j]]:  
 true\_len = true\_len + 1 #当前用户的实际已点菜品中存在推荐的菜品，每有一个，准确度+1  
 test\_emp\_accuracy[test\_emp\_id[j]] = true\_len/interest\_len  
print('测试集用户的推荐准确度字典',test\_emp\_accuracy)  
all\_accuracy = 0 #计算模型整体准确度，为所有测试集用户推荐准确度的均值  
for j in range(len(test\_emp\_id)):  
 all\_accuracy = all\_accuracy + test\_emp\_accuracy[test\_emp\_id[j]]  
all\_accuracy = all\_accuracy/len(test\_emp\_id)  
print('模型总体准确率为：',all\_accuracy)