**任务一：数据获取与探索分析**

**1.1 读取菜单数据并处理菜单字符数据**

首先，应用pandas包读取订单详情表（meal\_order\_detail.xlsx）中数据，并命名为data，并将data中的“dishes\_name”这一列数据读取出来，可以发现菜品名字数据中多次出现'\n', '\r'的后缀字符串。应用replace语句，将'\n', '\r'字符除去，效果如下图所示。

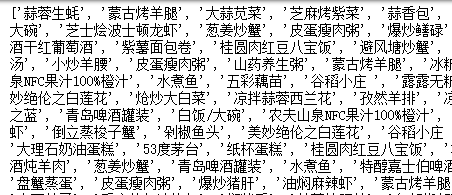


图1

由图1可见，已经除去了'\n', '\r'的后缀字符串。可以进入下一步分析。

**1.2 构建热销度评分指标**

热销度是在一定时间内产品销量的程度，本文根据餐饮企业近30天（2016.8.1-2016.8.31）进行每个菜品的热销度统计，计算公式如下所示：

其中，为某个菜品的销售份数， 为该时间段内所有菜品中的最大销售份数， 为该时间段内所有菜品中的最小销售份数。

应用上述计算公式计算得到每个菜品热销度，记为列r。效果如下图所示。



图2

**1.3 绘制销量前10条形图**

将列r进行降序排列，并选择排名前10位的菜品信息，调用matplotlib中pyplot模块进行条形图绘制。效果如下图所示。

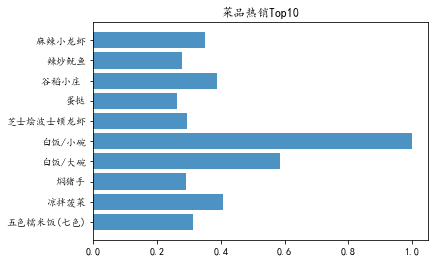


图3

由图3所示，“白饭/小碗”与“白饭/大碗”分别为热销度最高与次高。除此之外，菜品“麻辣小龙虾”，“谷稻小庄”，“凉拌菠菜”热销度都比较高。在2016年8月中，顾客在主食上偏好于白饭、糯米饭，在素菜上偏好于凉拌菠菜，在肉食上偏好于麻辣小龙虾、辣炒鱿鱼、波士顿龙虾、焖猪手，在甜食上偏好于蛋挞，在饮料上偏好于谷稻小庄。

**任务二：数据预处理**

**2.1 统计订单状态占比**

导入订单表（meal\_order\_info.csv），并记为data1。

利用函数value\_counts()将订单状态（order\_status）进行频率统计分析。

表1 订单状态频率统计

|  |  |
| --- | --- |
| 订单状态 | 频率 |
| 1 | 0.987302 |
| 0 | 0.009524 |
| 2 | 0.003175 |

表1为统计结果。其中订单状态“1”表示订单正常，其他均为异常情况。可见，正常状态下的订单占比为98.73%，有必要对订单进行异常状态剔除处理。

**2.2 选取有效的订单数据**

第一步，删除data1中的订单异常数据，即删除订单状态为“0”和“2”的数据。

第二步，删除含有“白饭”订单数据。因为“白饭”为常见食品，为了后续模型的稳健性，本文将“白饭”订单数据作删除处理。

第三步，提取重要特征。由于后续模型需要用到“emp\_id”以及“dishes\_name”信息，因此将这两列数据提取出来，并记为data3。data3数据如图4所示。

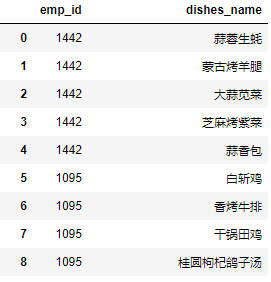


图4 data3

**任务三：构建模型准备**

**3.1 划分训练集与测试集**

**3.1.1** **只保留点了3个菜以上的客户**

由于点菜太少会影响模型效果，于是本文只保留点了3个菜以上的客户。本文对data3数据作根据客户ID（“emp\_id”）聚合处理，并应用numpy包中的函数count\_nonzero来统计每个ID所点的菜品总数，再根据菜品总数进行升序处理，效果如下图所示。

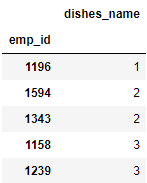


图5

如图5所示，客户ID号为1196，1594，1343的客户点的菜数量不够3个。于是，**将客户ID设置为索引号，并命名数据集为data4**,在data4中将索引号为1196，1594，1343的客户订单作删除处理，再对聚合排序，效果如下图所示。

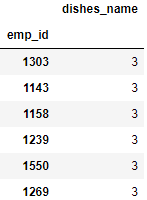


图6

可见，所有客户中最小订单数为3.

**3.1.2 将data4数据集划分为训练集和测试集**

由于sklearn模块中训练集与测试集分割函数的局限性，本文自定义训练集与测试集分割函数，命名为trainTestSplit(X,test\_size)。其中X为数据集，test\_size为测试集所占的比例。

在trainTestSplit(X,test\_size)函数中，首先进行对数据集中的索引号作去重处理并记录去重后数据长度，即为用户总数。然后通过test\_size计算出训练集、测试集的长度。调用random包中的sample函数，对去重后的索引号（即为用户ID）随机抽样，并把随机抽样结果记录在列表test\_index中。当test\_index长度达到测试集长度后，把剩余的索引号记录在列表train\_index中。根据列表train\_index的索引号，将该索引号在数据集所对应全部信息记为训练集。同理，根据列表test\_index的索引号，将该索引号在数据集所对应全部信息记为测试集。

调用trainTestSplit(X,test\_size)函数，设置数据集为data4，测试集比例为0.3，将data4数据集划分为训练集和测试集，效果如下所示。



图7 训练集与测试集

由图7可见，训练集和测试集分别记录了所对应客户的所有订单信息。

**3.2 构建客户-菜品二元矩阵**

本文构建新的数据框，命名为data5，其中data5的索引号为data4的去重后的索引号（即为去重后的客户ID），列为所有菜品名。共有225行143列。

遍历data4中的数据，若A客户点过菜品B，则在data5中的索引号为A，列名为B的对应数据框记为1；A客户未点菜品C，则在data5中的索引号为A，列名为C的对应数据框记为0。效果如下图所示。

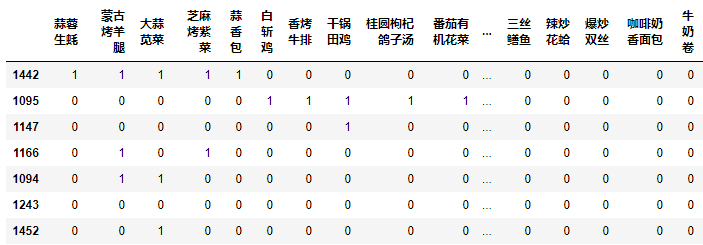


图8 客户-菜品二元矩阵data5

根据自定义函数trainTestSplit(X,test\_size)，设置数据集为data5，测试集比例为0.3，将data5划分为训练集与测试集，分别命名为数据集train，test。

查询可知，训练集客户数为158，测试集客户数为67。

**任务四：基于物品得协同过滤算法**

**4.1 计算相似度**

针对训练集的数据，利用Jaccard相似度计算公式，计算各个菜品之间的相似度。

其中，为表示训练集中菜品i， 为同时点了菜品i和菜品j的客户数，为同时点了菜品i或是菜品j的客户数。J表示菜品i和菜品j的相似度，取值范围为[0,1]。

根据上述计算公式，得到训练集内各个菜品之间的相似度矩阵，并将同一个菜品之间的相似度设为0。记菜品相似度矩阵为data7。



图9 训练集菜品相似度矩阵data7

**4.2 生成推荐列表**

新建数据框test\_data，索引号为data4的去重后的索引号（即为去重后的客户ID），列名为所有菜品名，并用0来初始化test\_data。

针对测试集的数据，对任意一个客户A，根据公式：

其中,P表示客户对所有菜品的感兴趣程度，SIM表示所有菜品之间的相似度，R表示客户用餐与否，R的取值为0或1。

根据菜品相似度矩阵（data7）知道SIM信息，找到客户A所点的菜品名，计算得到他对所有菜品的感兴趣程度。并且，根据菜品名的迭代，**将相同菜品计算得到的相似度做相加处理**。例如：客户A点了菜品B、菜品C，未点菜品D，菜品B与菜品D相似度为0.1，菜品C与菜品D相似度为0.2，则可以计算得客户A对菜品D的感兴趣度为0.3.

将所有客户对所有菜品的感兴趣程度的记录在test\_data中，test\_data即为所有客户的推荐菜品矩阵，效果如下图所示。



图10 推荐菜品矩阵test\_data

**任务五：模型评价**

**5.1 构建测试集****客户IP-菜品字典**

本文利用数据集test\_data，来构建建测试集客户IP-菜品字典。

首先，新建字典dict2,dict2中的键值为测试集的客户IP，并用空列表[]对dict2中所有键值所对应的值进行初始化。

其次，对测试集中任意一个客户A，应用rank函数，将该客户菜品感兴趣程度作降序处理，并且把前10个菜品名记录为dict2[A]所对应的值。

通过遍历全部客户IP，得到以下客户IP-菜品字典。

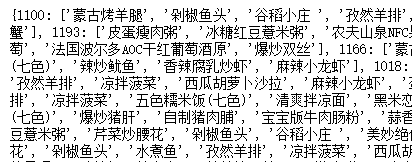


图11 客户IP-推荐菜品字典

**5.2 构建评价指标**

首先，构建测试集的客户原始菜单字典dict1，对任意一个客户A，将其点过的所有菜名记录为dict1[A]所对应的值。dict1如下所示。

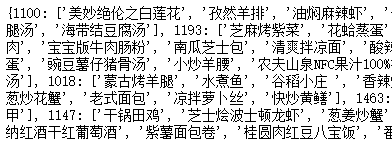


图12 客户IP-原始菜品字典

其次，新建数据框p\_data,其索引值为测试集客户ID，列名为p,记录每个用户对应的推荐准确率，推荐准确率p公式为：

其中N1为正确推荐菜品数，即为全部推荐菜名中包含了客户点过的菜名的数量。N2为总推荐菜品数，即为全部推荐菜名的数量。

分别计算出全部客户的p值，并作简要统计分析。

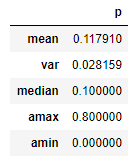


图13 p值统计分析

可见全部客户的p值均值为0.12，中位数为0.1，最大值为0.8。

调用matplotlib中pyplot模块，以测试集客户ID为X轴，p值为Y轴，作散点图如下所示。

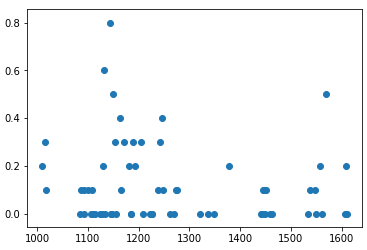


图14 p值散点图

可见绝大部分数据分布在0-0.4之间，p值为0的点较多。为了探讨每个组别（“0-0.2”，“0.2-0.4”，“0.4-0.6”，“0.6-0.8”，“0.8-1.0”）的数据点数目，本文将数据点作分组处理，并贴上相应组别标签，再计算每个组别的数据点数目。

表2 组别频数统计

|  |  |
| --- | --- |
| 组别 | 频数 |
| 0-0.2 | 48 |
| 0.2-0.4 | 13 |
| 0.4-0.6 | 4 |
| 0.6-0.8 | 1 |
| 0.8-1.0 | 1 |

综上所述，用户的推荐准确率相对而言不高，绝大部分都在[0,0.4]范围内，有极少部分准确率比较高，能高于0.6。