|  |  |
| --- | --- |
|  |  |
| 项目报告C |  |
|  |  |
|  | 课程：数据挖掘姓名：姜其升 |
|  | 学号：1752058指导教师：饶卫雄教授 |

[项目报告C 1](#_Toc39137296)

[1 算法概述 3](#_Toc39137301)

[1.1 FTCTree距离的计算 3](#_Toc39137302)

[1.1.1 FTCTree的构建 3](#_Toc39137303)

[1.1.2 交集树（IR）和并集树（UR）的构建 3](#_Toc39137304)

[1.1.3 FTCTree距离的计算 3](#_Toc39137305)

[1.2 簇的质心计算 3](#_Toc39137306)

[1.3 聚类过程 4](#_Toc39137307)

[1.3.1 BIC的计算 4](#_Toc39137308)

[1.3.2 使用K-means算法分裂 4](#_Toc39137309)

[1.4 聚类结果评价 4](#_Toc39137310)

[2 关键代码 5](#_Toc39137311)

[3 聚类结果及运行截图 6](#_Toc39137312)

[4 聚类结果分析 9](#_Toc39137313)

[4.1 距离分布情况 9](#_Toc39137314)

[4.2 SC/CP的值 9](#_Toc39137315)

[5 算法性能 10](#_Toc39137316)

## 1 算法概述

### 1.1 FTCTree距离的计算

#### 1.1.1 FTCTree的构建

将数据中的最近的时间作为时间分类评级的当前时间，将时间进行评级，如下表所示。

|  |  |
| --- | --- |
| 时间 | Level |
| 购买时间 <= 1个月 | 4 |
| 1个月 < 购买时间 <= 2个月 | 3 |
| 1个月 < 购买时间 <= 4个月 | 2 |
| 购买时间 > 4个月 | 1 |

根据顾客购买的商品种类和购买时间等因素为每位顾客构建FTCTree。

#### 1.1.2 交集树（IR）和并集树（UR）的构建

交集树：相同节点保留，并将频率相加。

并集树：相同节点频率相加并保留，不同节点全部保留。

#### 1.1.3 FTCTree距离的计算

其中，，，l为交集树的子路径除了root的长度，L为并集树的深度。

### 1.2 簇的质心计算

本题中簇内树的质心采用以下算法进行计算。

1. Input: C: a cluster of FTCTrees
2. Output: ct: centroid tree of the cluster
4. utree = unionTree(C);
5. freq = 1;
6. mindist=+∞;
7. freqStep = AvgFreq(utree);
8. freqEnd = MaxFreq(utree);
9. WHILE freq≤freqEnd DO:
10. utree = update(utree, f req);
11. dist = ;
12. IF dist < mindist THEN
13. mindist = dist;
14. ct = utree;
15. END IF
16. freq = freq + freqStep;
17. END WHILE
18. **return** ct ;

### 1.3 聚类过程

本题中，我们首先假设所有的FTCTree属于同一个簇，之后在每次迭代中计算BIC，之后用K = 2的K-means算法，将初始簇分裂为两个簇，再计算BIC值，如果分裂后的BIC大于分裂前的BIC，那么就进行分裂，否则不分裂。在簇中个数小于等于2时不再进行分裂。最后没有簇可以分裂时结束。最终结果即为聚类结果，无需初始参数。

#### 1.3.1 BIC的计算

表示簇及其质心。表示likelihood。和表示该簇内用户个数以及该簇包含的商品数（用户一共买过多少种商品，以最细粒度计算），簇的个数（分裂前为1，分裂后为2），表示该簇内用户个数。表示用户树和质心树之间距离的variance.

该公式用于计算一个簇是否分裂为两个簇。在公式中，如果是分裂前，只是待分裂的簇。分裂后，表示分裂后的两个簇。

#### 1.3.2 使用K-means算法分裂

设定K= 2，在分裂时，首先选取距离最远的两个点分别作为两个簇的初始元素计算质心。 之后进行k-means聚类，聚类结果即为分裂后的结果。

1. FOR 每个剩余的点p DO:
2. 找到离点p最近的质心;
3. 把点p加入到这个质心所在的簇中;
4. 调整这个簇的质心;
5. END;

### 1.4 聚类结果评价

我们这里采用Silhouette Coefficient（SC）和Compactness（CP）进行评价。

计算时，样本间距离用FTCTree距离进行计算。

其中a(i)为样本i与同一簇内其他样本的平均距离，b(i)为样本i与不同簇的其他所有样本的平均距离。SC描述了簇的内聚情况，与其他簇的边界是否清晰，越接近与1，效果越好。

其中ct为每个簇的质心，CP描述了簇每个成员距离质心的平均距离，一个较小的CP值通常意味着更好的聚类效果。

## 2 关键代码

1. # 计算距离矩阵
2. matrixPD = pd.DataFrame(data=np.matlib.zeros((len(userToTree),len(userToTree))),
3. index=userGroup, columns=userGroup)
4. **for** i **in** range(len(userGroup)):
5. **for** j **in** range(i+1, len(userGroup)):
6. matrixPD.loc[userGroup[i], userGroup[j]] = matrixPD.loc[userGroup[j], userGroup[i]] = Dist(userToTree[userGroup[i]], userToTree[userGroup[j]])
8. # 进行分裂
9. **while** len(waitingDeque) != 0:
10. userListBefore = waitingDeque.popleft()
11. ctBefore = GetCT(userListBefore)
12. bicBefore = Bic(userListBefore, ctBefore)
13. kmeans = kmeans2(userListBefore)
14. bicAfter = Bic(kmeans[0], kmeans[1])
15. **if** bicAfter >= bicBefore:
16. **for** i **in** kmeans[0]:
17. **if** len(i) > 2:
18. waitingDeque.append(i)
19. **else**:
20. finshDeque.append(i)
21. **else**:
22. finshDeque.append(userListBefore)
23. # 计算sc
24. pointNum = len(finshDeque)
25. **for** i **in** range(pointNum):
26. **if** len(finshDeque[i]) == 0:
27. **continue**
28. **for** customerId **in** finshDeque[i]:
29. sc = sc/len(userGroup)
31. # 计算cp
32. **for** i **in** range(pointNum):
33. cp = cp + cpi/len(finshDeque[i])
34. cp = cp/pointNum

## 3 聚类结果及运行截图

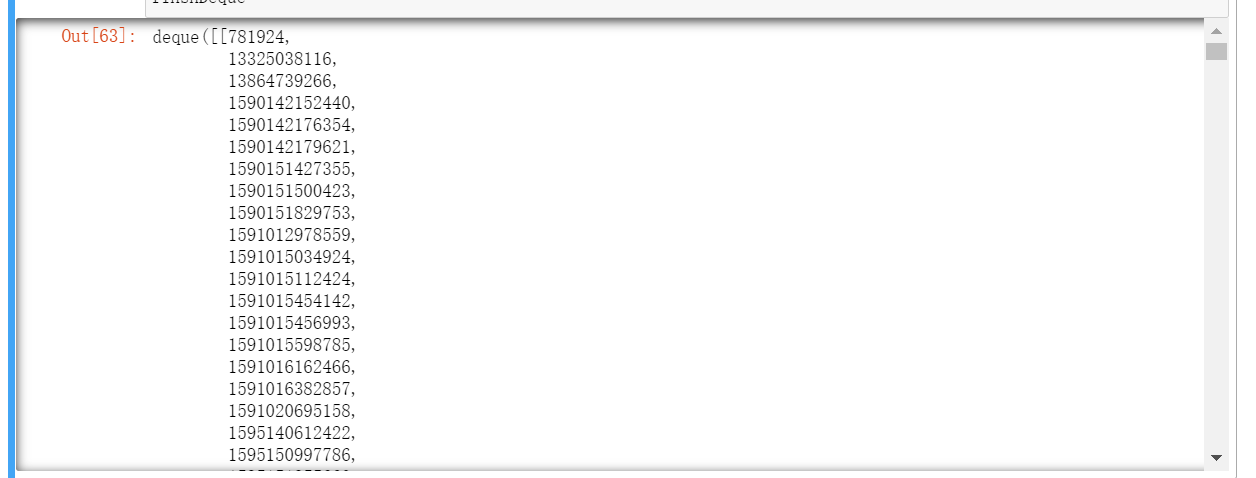
K = 4

SC = 0.189

CP = 0.289

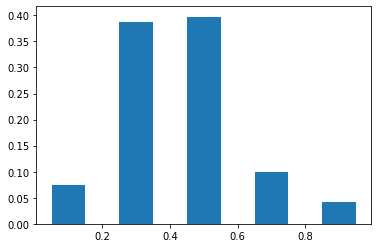
具体结果：

[781924,13325038116,13864739266,1590142152440,1590142176354,1590142179621,1590151427355,1590151500423,1590151829753,1591012978559,1591015034924,1591015112424,1591015454142,1591015456993,1591015598785,1591016162466,1591016382857,1591020695158,1595140612422,1595150997786,1595151355660,1595151630774,1598140055923,2900000060012,2900000138223,2900000254503,2900000304376,2900000384187,2900000574274,2900000623040,2900000776104,2900001222099,2900001240246,2900001582933,2900001645768,2900002944495,2900003116341]  
[2900001137843,2900000780835,2900000855502]  
[1593140971419,13854627199,15954611837,15963883482,15963885355,18554652702,18654692914,18764569988,1590120464497,1590120718170,1590130640102,1590130817948,1590140304209,1590140304506,1590140305107,1590140306678,1590140307286,1590140307415,1590140307767,1590140307859,1590140308030,1590140308214,1590141216259,1590141414228,1590142128124,1590142129763,1590142148962,1590142149112,1590142149853,1590142150507,1590142151689,1590142176521,1590142191722,1590142194327,1590142197076,1590142201148,1590142201162,1590142202190,1590142203104,1590142203180,1590142203401,1590142204569,1590142205993,1590142206747,1590142210195,1590142213189,1590142213356,1590142213981,1590142215619,1590142230179,1590142242370,1590142456470,1590142477895,1590142514866,1590142516068,1590142516600,1590142517836,1590142518307,1590142519632,1590142520300,1590142648356,1590142751995,1590150321449,1590151076256,1590151103907,1590151210391,1590151321240,1590151363066,1590151428581,1590151432557,1590151459721,1590151464053,1590151467962,1590151472027,1590151472782,1590151682891,1590151825175,1590151832586,1590151832593,1590151835808,1590151842899,1590151843124,1590151977348,1591013167365,1591013227632,1591013471721,1591013766650,1591013866800,1591013877134,1591014083046,1591014130276,1591014133086,1591014214433,1591014255948,1591014278527,1591014338276,1591014523610,1591014577972,1591014637324,1591014680184,1591014943968,1591015027858,1591015074944,1591015088262,1591015097233,1591015097516,1591015136338,1591015184407,1591015218485,1591015269418,1591015327590,1591015408602,1591015419448,1591015420000,1591015452933,1591015454210,1591015454562,1591015454692,1591015454814,1591015456955,1591015457273,1591015476700,1591015478124,1591015480455,1591015480745,1591015483951,1591015500993,1591015501716,1591015506575,1591015519216,1591015521639,1591015600815,1591015604417,1591016154157,1591016159916,1591016170300,1591016172045,1591016174353,1591016174957,1591016310287,1591016320026,1591016350504,1591016411137,1591016439575,1591016440328,1591016443121,1591016494765,1591016517778,1591016517921,1591016519031,1591020269281,1591020377344,1591020742401,1591020742449,1591030092053,1591030217500,1591030411823,1591040087452,1591040155274,1591040161114,1591040203906,1591040356657,1591040416658,1591040462983,1591040497398,1592010017509,1592013087615,1592014016171,1592015013285,1592015016880,1592015018662,1592015026841,1592015046818,1592015053267,1592015054592,1592015056299,1592015056343,1592015058774,1592015058781,1592015059993,1593141148827,1593150463515,1593160622155,1594140460286,1595112593216,1595130081566,1595140489499,1595150263683,1595150403423,1595150738976,1595150770945,1595150771652,1595150883423,1595150992095,1595151123344,1595151168727,1595151204906,1595151442803,1595151609213,1595151616471,1595151629143,1595151630446,1595151631146,1595151738968,1595151786686,1595151866807,1595151915963,1595160035249,1595160353664,1596130424193,1596140586607,1596140587802,1596140594374,1596140628703,1596160006376,1596160024097,1596160058337,1598140107578,1598140129341,2900000040663,2900000047648,2900000048812,2900000050785,2900000071742,2900000072022,2900000074163,2900000124974,2900000125261,2900000138056,2900000164567,2900000177079,2900000178106,2900000179417,2900000194298,2900000203457,2900000210790,2900000216952,2900000224001,2900000224315,2900000239876,2900000245396,2900000245914,2900000246638,2900000253230,2900000255463,2900000256637,2900000268692,2900000271814,2900000284333,2900000307377,2900000308312,2900000317307,2900000330863,2900000337633,2900000339637,2900000342651,2900000345300,2900000350175,2900000350458,2900000353848,2900000372849,2900000383913,2900000386440,2900000386495,2900000394940,2900000401495,2900000404731,2900000408586,2900000416307,2900000417663,2900000418073,2900000448520,2900000449732,2900000451452,2900000452756,2900000468030,2900000476042,2900000485990,2900000490130,2900000544765,2900000557161,2900000562165,2900000566378,2900000587649,2900000588691,2900000604360,2900000680036,2900000780743,2900000781603,2900000784918,2900000840164,2900000840805,2900000845121,2900000847224,2900000849785,2900000852525,2900000863286,2900000863880,2900000864498,2900000870994,2900000874145,2900000874930,2900000880467,2900000883475,2900000883819,2900000884007,2900000890688,2900000934474,2900000936997,2900000941830,2900001020541,2900001047739,2900001050364,2900001050500,2900001050791,2900001067881,2900001067911,2900001068499,2900001068833,2900001078658,2900001090605,2900001092012,2900001093019,2900001094702,2900001095952,2900001109598,2900001157902,2900001211239,2900001211321,2900001215688,2900001216616,2900001220255,2900001234399,2900001238632,2900001255080,2900001270199,2900001270205,2900001307901,2900001308168,2900001309639,2900001331616,2900001355049,2900001364058,2900001398121,2900001413886,2900001419826,2900001421652,2900001423069,2900001431651,2900001432658,2900001432672,2900001435116,2900001436120,2900001436366,2900001436410,2900001436700,2900001437165,2900001437424,2900001451932,2900001452366,2900001462648,2900001463379,2900001465380,2900001465816,2900001465960,2900001466509,2900001467353,2900001470780,2900001497336,2900001509459,2900001538596,2900001540117,2900001540971,2900001550246,2900001559720,2900001568630,2900001575201,2900001578363,2900001661508,2900001663823,2900001668996,2900001671408,2900001684507,2900002532630,2900002542837,2900002548457,2900002701746,2900002927733,2900002930092,2900002932416,2900002932614,2900002933024,2900002933918,2900002934304,2900002934342,2900002934762,2900002934892,2900002934915,2900002936520,2900002936940,2900002937428,2900002937503,2900002944402,2900002944471,2900002969559,2900003105154,2900003105161,2900003106281,2900003106502,2900003107363,2900003108773,2900003110479,2900003114002,2900003114170,2900003114613,2900003114880,2900003115009,2900003116822,2900003117386,2900003120607,2900003122076,2900003123479,6222021615010082737,6227002180901670266,6227002180921663895]  
[1590142201513,15954688237,1590142240482,1590151457185,1591015091286,1594140121125,1595150738747,1595151375125,1595151644528,2900000050419,2900000072701,2900000192492,2900000348561,2900000385627,2900000390256,2900000516892,2900000598973,2900000780040,2900000856585,2900001053938,2900001270014]



## 4 聚类结果分析

### 4.1 距离分布情况



上图为FTCTree距离的分布情况，我们可以看到，FTCTree距离的分布较为分散，大部分的顾客分布在0.2~0.6的区间之内，占比超过70%，且在这两个区间内的分布情况也较为平均。在其余区间内的分布也较为平均，每个占比不到10%。按照此方法计算顾客相似度适中，有梯度，分布较为松散，有利于聚类，可以得到较好的聚类结果，区分度较好。

### 4.2 SC/CP的值

|  |  |
| --- | --- |
| SC | 0.189 |
| CP | 0.289 |

通过计算，我们可以得到上述的SC和CP值，与之前的聚类结果想比较，SC和CP值都较为优越，效果比较理想，但是结果粒度过大，区分不够细腻，可能存在其他原因。

## 5 算法性能

算法性能较优，不会存在过多迭代的情况，也不需要指定初始值就可以进行聚类。在计算FTCTree距离时，只要采用较为合适的数据结构就能较快的计算出结果。比如，我们在存放时，使用的时py自带的dict（字典）数据结构，只要将算法做相应的对应，变换形式，就可以达到使用树形数据结构存储相同的功能，而不需要进行额外的深度或广度迭代遍历，加速了树的构建和距离的计算。树的结构定义如下：

1. **class** DictFTCTree():
3. **def** \_\_init\_\_(self, custProd = 1, dictTreeNodes = 1):
4. **if** type(dictTreeNodes) != int:
5. self.nodes = dictTreeNodes
6. **elif** type(custProd) != int:
7. self.nodes = {}
8. self.create(custProd)
9. **else**:
10. self.nodes = {}
12. **def** create(self, custProd):
13. custProd = custProd.loc[:,["pluno", "level"]].sort\_values("pluno")
14. **for** i **in** custProd.itertuples():
15. **for** j **in** range(i.level):
16. pluno = int(str(i.pluno)[:j+2])
17. **if** pluno **in** self.nodes.keys():
18. self.nodes[pluno] = self.nodes[pluno] + 1
19. **else**:
20. self.nodes[pluno] = 1
21. **def** show(self):
22. **print**(self.nodes)
24. **def** union(self, other):
25. newNodes = self.nodes.copy()
26. **for** i **in** other.nodes.keys():
27. **if** i **in** newNodes.keys():
28. newNodes[i] = newNodes[i] + other.nodes[i]
29. **else**:
30. newNodes[i] = other.nodes[i]
31. **return** DictFTCTree(dictTreeNodes=newNodes)
33. **def** intersection(self, other):
34. newNodes = {}
35. **for** i **in** self.nodes.keys():
36. **if** i **in** other.nodes.keys():
37. newNodes[i] = self.nodes[i] + other.nodes[i]
38. **return** DictFTCTree(dictTreeNodes=newNodes)