杭州电子科技大学

《数据挖掘课程设计》

课 程 设 计 报 告

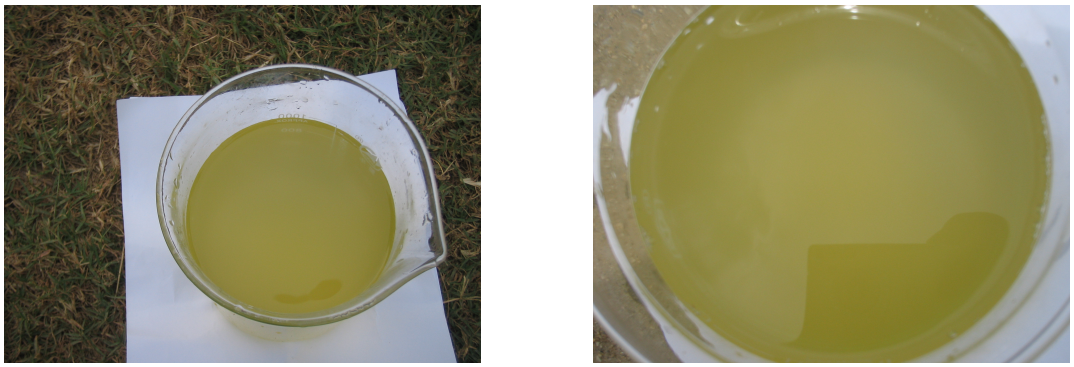
专　　业	信息与计算科学
班　　级	19071232
学生姓名	张艺泷
学　　号	19071232
指导教师	邵新平
实验地点	6 教 402
完成日期	2022 年 6月 29日

目录

1. 水质图片分类

1.1 数据集介绍

该数据集为某湖水样本，共 203 张图片，由类别 + 下划线 + 序号命名。



(a) (b)

图 1　数据集

1.2 数据预处理

1.2.1 图片裁剪

为了提取有效信息，规范数据输入，只裁剪中间含有有水质样本的图像。保留中心 100×100 的图像。

图 2　裁剪后

1.2.2 计算颜色矩

获取裁剪的图像后，对其 RGB 三个通道分别计算一到三阶颜色矩。之后采用 str.split 提取图片的类别，并将其存入 Dataframe

1.3 基于 SVM 的水质图片分类模型

对于此类小样本，采用 SVM 模型对其进行训练。

首先将预处理好的数据分割为训练集与测试集。之后调用 sklearn.svm 进行模型训练。

1.4 模型结果

采取混淆矩阵对训练结果进行模型评价。可以看到 SVM 模型对 1,2,3 类水质预测较好。4,5 类为不平衡数据。

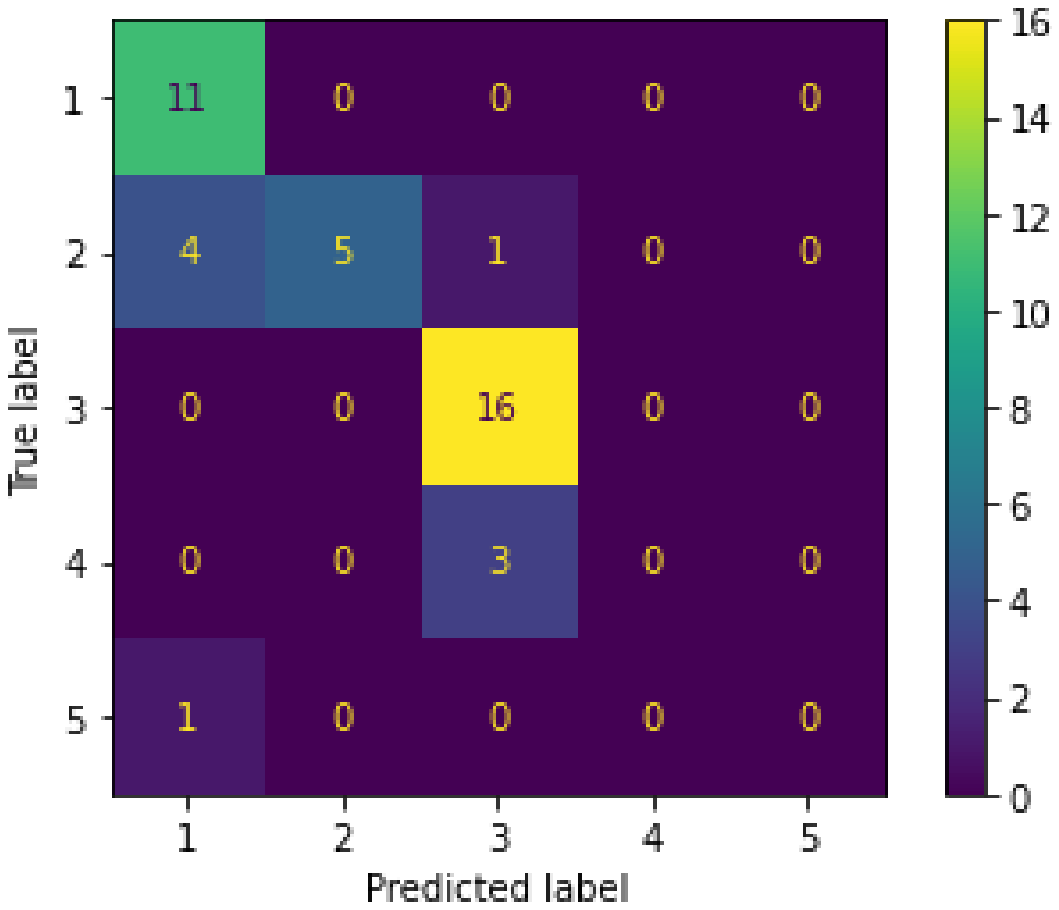


图 3　测试集混淆矩阵

2. 航空公司客户价值分析

2.1 数据集介绍

该数据为某航空公司客户信息，包含会员档案信息和其乘坐航班记录等信息。

LOAD_TIME	FFP_DATE	LAST_TO_END	FLIGHT_COUNT	SEG_KM_SUM	AVG_DISCOUNT
2014/4/1	2006/03/31	6.6	3	18770	0.66
2014/4/1	2006/03/31	3.8	24	35087	0.62
2014/4/1	2006/04/07	2.8	9	20660	0.52
2014/4/1	2006/08/10	1	12	23071	0.51
2014/4/1	2008/02/07	3.17	3	2897	0.95
2014/4/1	2010/09/16	1.57	3	4608	0.65
2014/4/1	2011/04/28	17.83	2	3390	0.48
2014/4/1	2012/03/09	4.13	8	11797	1.35
2014/4/1	2012/08/31	5.9	6	6355	0.75
2014/4/1	2005/09/29	0.4	54	62170	0.79
2014/4/1	2009/01/23	2.33	24	15894	0.6
2014/4/1	2009/01/30	0.07	13	19517	0.72
2014/4/1	2009/01/30	4.63	10	12686	0.55
2014/4/1	2009/02/13	14.93	13	10992	1.33
2014/4/1	2009/04/25	10.27	3	4137	0.67
2014/4/1	2009/06/12	0.33	19	37415	0.63
2014/4/1	2010/03/25	1.63	13	24156	0.79
2014/4/1	2010/04/15	22.23	3	1559	0.87
2014/4/1	2010/05/20	23.17	2	1870	0.6
2014/4/1	2010/07/08	2.6	30	46621	0.93
2014/4/1	2011/03/10	4.57	4	7999	0.58

图 4　数据信息

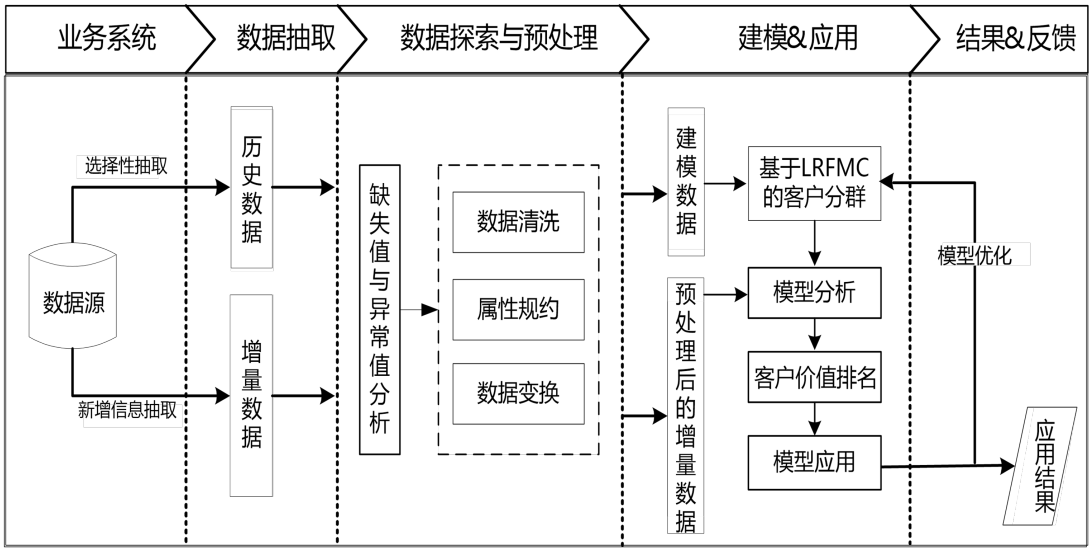


图 5　流程图

2.2 数据预处理

首先，以 2014-03-31 为结束时间，选取宽度为两年的时间段作为分析观测窗口，抽取观测窗口内有乘机记录的所有客户的详细数据形成历史数据。对于后续新增的客户详细信息，利用其数据中最大的某个时间点作为结束时间，采用上述同样的方法进行抽取，形成增量数据。根据末次飞行日期，从航空公司系统内抽取 2012-04-01 至 2014-03-31 内所有乘客的详细数据，总共 62988 条记录

原始数据中存在票价为空值，票价为空值的数据可能是客户不存在乘机记录造成。票价最小值为 0、折扣率最小值为 0、总飞行公里数大于 0 的数据。其可能是客户乘坐 0 折机票或者积分兑换造成。

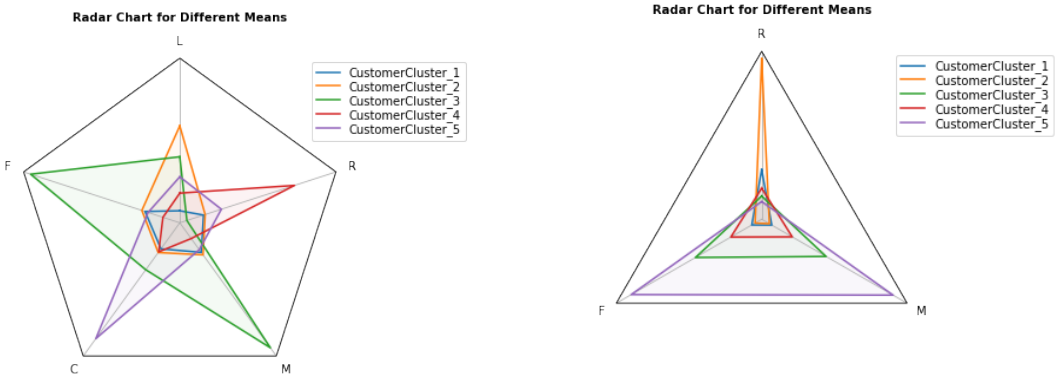
原始数据中属性太多，根据 LRFMC 模型，选择与其相关的六个属性，删除不相关、弱相关或冗余的属性。

2.3 基于 K-means 的客户聚类

采用 K-Means 聚类算法对客户数据进行分群，将其聚成五类

2.4 模型结果

对聚类结果进行特征分析，其中客户群 1 在 F、M 属性最大，在 R 属性最小；客户群 2 在 L 属性上最大；客户群 3 在 R 属性上最大，在 F、M 属性最小；客户群 4 在 L、C 属性上最小；客户群 5 在 C 属性上最大。



(a) (b)

图 6　结果可视化

3. 基于用户的协同过滤

3.1 数据集说明

该数据包括 IMDB 用户对不同电影的打分情况，分为用户表，电影表，用户看过的电影及打分情况表。其中，打分情况表共 100000 行。

3.2 相关系数的计算

实现基于用户的协同过滤算法第一个重要的步骤就是计算用户之间的相似度。而计算相似度，建立相关系数矩阵目前主要分为以下几种方法。

3.2.1 皮尔森相关系数

皮尔逊相关系数一般用于计算两个定距变量间联系的紧密程度，它的取值在 [-1，+1] 之间。用数学公式表示，皮尔森相关系数等于两个变量的协方差除于两个变量的标准差。计算公式如下所示：

$$s(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

由于皮尔逊相关系数描述的是两组数据变化移动的趋势，所以在基于用户的协同过滤系统中，经常使用。描述用户购买或评分变化的趋势，若趋势相近则皮尔逊系数趋近于 1，也就是我们认为相似的用户。

3.2.2 基于欧几里德距离的相似度余弦相似度

欧几里德距离计算相似度是所有相似度计算里面最简单、最易理解的方法。计算出来的欧几里德距离是一个大于 0 的数，为了使其更能体现用户之间的相似度，可以把它规约到 (0,1] 之间，最终得到如下计算公式

$$s(X,Y) = \frac{1}{1 + \sum \sqrt{(X_i - Y_i)^2}} \tag{2}$$

只要至少有一个共同评分项，就能用欧几里德距离计算相似度；如果没有共同评分项，那么欧几里德距离也就失去了作用。其实照常理理解，如果没有共同评分项，那么意味着这两个用户或物品根本不相似。

3.2.3 余弦相似度

余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。余弦相似度更加注重两个向量在方向上的差异，而非距离或长度上。计算公式如下所

示：

$$s(X,Y)=\cos\theta=\frac{\vec{x}*\vec{y}}{\|x\|*\|y\|} \tag{3}$$

3.3 基于用户的协同过滤

基于用户的协同过滤算法，另一个重要的步骤就是计算用户 u 对未评分商品的预测分值。首先根据上一步中的相似度计算，寻找用户 u 的邻居集 N□U, 其中 N 表示邻居集，U 表示用户集。然后，结合用户评分数据集，预测用户 u 对项 i 的评分，计算公式如下所示:

$$p_{u,i}=\bar{r}+\frac{\sum_{u'\subset N}s\left(u-u'\right)\left(r_{u',i}-\bar{r}_{u'}\right)}{\sqrt{\sum_{u'\subset N}\left|s\left(u-u'\right)\right|}} \tag{4}$$

其中，s(u- u') 表示用户 u 和用户 u' 的相似度。

3.4 算法的实现

现有的部分电影评分数据如下表:

	UserId	MovieId	Rank	No
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

图 7　部分数据

实现代码如下所示:

```
def similarity(userid1:int,userid2:int,data:pd.DataFrame):
# 获取userid1的电影id 并将其作为索引
movie_index = list(data['MovieId'][data['UserId']==userid1].value_counts().index)

#创建dataframe
join_dataframe = pd.DataFrame(index=movie_index)

#获取数据
data1 = data[['MovieId','Rank']][data['UserId']==userid1].set_index('MovieId')
data2 = data[['MovieId','Rank']][data['UserId']==userid2].set_index('MovieId')
```

```
#换列名
data1.columns = ['userid1']
data2.columns = ['userid2']

#join 连接
join_dataframe = join_dataframe.join(data1)
join_dataframe = join_dataframe.join(data2)
join_dataframe = join_dataframe.fillna(0)

# 计算相关系数
norm1=np.sqrt(np.dot(join_dataframe['userid1'],join_dataframe['userid1']))
norm2=np.sqrt(np.dot(join_dataframe['userid2'],join_dataframe['userid2']))
val=np.dot(join_dataframe['userid1'],join_dataframe['userid2'])/(norm1*norm2)

return val

c_user = 93
s = {}
all_user_id = data['UserId'].unique()

for cid in all_user_id:
s[cid] = similarity(cid,c_user,data)
s[c_user] = 0
Max = max(s.items(),key=lambda x:x[1] if x[1]<=1 else 0) # items以列表返回可遍历的(键, 值)
    元组数组

Max = max(s.items(),key=lambda x:x[1] if x[1]<=1 else 0) # items以列表返回可遍历的(键, 值)
    元组数组
```

3.5 模型结果

图 8

可以看到，与用户 20 最相似的为 769，相似度为 0.59，因此可以对 20 推荐 14, 125, 151, 275, 276, 283, 412, 477, 815, 820, 845, 866 电影。

4.