分析并预处理user_profile数据集

缺失值处理方案：

1. 填充方案：结合用户的其他特征值，利用随机森林算法进行预测；但产生了大量人为构建的数据，一定程度上增加了数据的噪音
2. 把变量映射到高维空间：如pvalue_level的1维数据，转换成是否1、是否2、是否3、是否缺失的4维数据；这样保证了所有原始数据不变，同时能提高精确度，但这样会导致数据变得比较稀疏，如果样本量很小，反而会导致样本效果较差，因此也不能滥用

最终方案2的效果好

In [ ]:

```
'''
特征选取
我该选取那些特征呢？
除了前面处理的pvalue_level和new_user_class_level需要作为特征以外，(能体现出用户的购买
分类特征值：个数
- cms_segid:  97
- cms_group_id:  13
- final_gender_code:  2
- age_level:  7
- shopping_level:  3
- occupation:  2
- pvalue_level
- new_user_class_level
- price
根据经验，以上几个分类特征都一定程度能体现用户在购物方面的特征，且类别都较少，都可以用
'''
```

In [1]:

```
import os
# 配置pyspark和spark driver运行时 使用的python解释器
JAVA_HOME = '/root/bigdata/jdk'
PYSPARK_PYTHON = '/miniconda2/envs/py365/bin/python'
# 当存在多个版本时，不指定很可能会导致出错
os.environ['PYSPARK_PYTHON'] = PYSPARK_PYTHON
os.environ['PYSPARK_DRIVER_PYTHON'] = PYSPARK_PYTHON
os.environ['JAVA_HOME'] = JAVA_HOME
# 配置spark信息
from pyspark import SparkConf
from pyspark.sql import SparkSession

SPARK_APP_NAME = 'preprocessingUserProfile'
SPARK_URL = 'spark://192.168.58.100:7077'

conf = SparkConf()
config = (
    ('spark.app.name',SPARK_APP_NAME),
    ('spark.executor.memory','2g'),
    ('spark.master',SPARK_URL),
    ('spark.executor.cores','2')
)
conf.setAll(config)

spark = SparkSession.builder.config(conf=conf).getOrCreate()
```

In [2]:
```
df = spark.read.csv('/data/user_profile.csv',header=True)
# 发现pvalue_level和new_user_class_level存在空值：
#（注意此处的null表示空值，而如果是NULL，则往往表示是一个字符串）
# 因此直接利用schema就可以加载进该数据，无需替换null值
df.show()
```

```
+------+---------+-----------+----------------+---------+-----------+--------------+
----------+--------------------+
|userid|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_level|
occupation|new_user_class_level |
+------+---------+-----------+----------------+---------+-----------+--------------+
----------+--------------------+
|   234|        0|           5|               2|        5|        null|             3|
0|                   3|
|   523|        5|           2|               2|        2|           1|             3|
1|                   2|
|   612|        0|           8|               1|        2|           2|             3|
0|                null|
|  1670|        0|           4|               2|        4|        null|             1|
0|                null|
|  2545|        0|          10|               1|        4|        null|             3|
0|                null|
|  3644|       49|           6|               2|        6|           2|             3|
0|                   2|
|  5777|       44|           5|               2|        5|           2|             3|
0|                   2|
|  6211|        0|           9|               1|        3|        null|             3|
0|                   2|
|  6355|        2|           1|               2|        1|           1|             3|
0|                   4|
|  6823|       43|           5|               2|        5|           2|             3|
0|                   1|
|  6972|        5|           2|               2|        2|           2|             3|
1|                   2|
|  9293|        0|           5|               2|        5|        null|             3|
0|                   4|
|  9510|       55|           8|               1|        2|           2|             2|
0|                   2|
| 10122|       33|           4|               2|        4|           2|             3|
0|                   2|
| 10549|        0|           4|               2|        4|           2|             3|
0|                null|
| 10812|        0|           4|               2|        4|        null|             2|
0|                null|
| 10912|        0|           4|               2|        4|           2|             3|
0|                null|
| 10996|        0|           5|               2|        5|        null|             3|
0|                   4|
| 11256|        8|           2|               2|        2|           1|             3|
0|                   3|
| 11310|       31|           4|               2|        4|           1|             3|
0|                   4|
+------+---------+-----------+----------------+---------+-----------+--------------+
----------+--------------------+
only showing top 20 rows
```

In [3]:
```
df.printSchema()
```

```
root
 |-- userid: string (nullable = true)
 |-- cms_segid: string (nullable = true)
 |-- cms_group_id: string (nullable = true)
 |-- final_gender_code: string (nullable = true)
 |-- age_level: string (nullable = true)
 |-- pvalue_level: string (nullable = true)
 |-- shopping_level: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- new_user_class_level : string (nullable = true)
```

In [7]:
```
# # 延申学习dropna()
# # str的空值不能通过dropna()去除掉
# print(df.select('pvalue_level').count())
# df.dropna()
# print(df.select('pvalue_level').count())
```

```
1061768
1061768
```

In [4]:
```
# 注意：这里的null会直接被pyspark识别为None数据，也就是na数据，所以这里可以直接利用sch
# 注意：如果数据集中存在NULL字样的数据，无法直接设置schema，只能先将NULL类型的数据处理
# 如果直接schema，下图会变成下下图：
# +----------+-------+-----------+--------+------+-----+
# |adgroup_id|cate_id|campaign_id|customer| brand|price|
# +----------+-------+-----------+--------+------+-----+
# |    375706|   4520|     387991|       6|  NULL| 99.0|

# +---------+------+----------+----------+-------+-----+
# |adgroupId|cateId|campaignId|customerId|brandId|price|
# +---------+------+----------+----------+-------+-----+
# |     null|  null|      null|      null|   null| null|
#处理方式如下：
# from pyspark.sql.types import IntegerType, FloatType
# ad_feature_df = df.\
#     withColumn("adgroup_id", df.adgroup_id.cast(IntegerType())).withColumnRenamed("a
#     withColumn("cate_id", df.cate_id.cast(IntegerType())).withColumnRenamed("cate_id
#     withColumn("campaign_id", df.campaign_id.cast(IntegerType())).withColumnRenamed
#     withColumn("customer", df.customer.cast(IntegerType())).withColumnRenamed("custo
#     withColumn("brand", df.brand.cast(IntegerType())).withColumnRenamed("brand", "br
#     withColumn("price", df.price.cast(FloatType()))
```

In [3]:
```
# 当然了，我们只能看到部分数值，所以有可能还有NULL，只是没有被我们看到，所以最好使用上
from pyspark.sql.types import StructType, StructField, IntegerType
schema = StructType([
    StructField("userId", IntegerType()),
    StructField("cms_segid", IntegerType()),
    StructField("cms_group_id", IntegerType()),
    StructField("final_gender_code", IntegerType()),
    StructField("age_level", IntegerType()),
    StructField("pvalue_level", IntegerType()),
    StructField("shopping_level", IntegerType()),
    StructField("occupation", IntegerType()),
    StructField("new_user_class_level", IntegerType())
])
user_profile_df = spark.read.csv('/data/user_profile.csv',header=True,schema=schema)
user_profile_df.printSchema()
user_profile_df.show()
```

```
root
 |-- userId: integer (nullable = true)
 |-- cms_segid: integer (nullable = true)
 |-- cms_group_id: integer (nullable = true)
 |-- final_gender_code: integer (nullable = true)
 |-- age_level: integer (nullable = true)
 |-- pvalue_level: integer (nullable = true)
 |-- shopping_level: integer (nullable = true)
 |-- occupation: integer (nullable = true)
 |-- new_user_class_level: integer (nullable = true)

+------+---------+------------+-----------------+---------+------------+-------------
-+----------+--------------------+
|userId|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_leve
l|occupation|new_user_class_level|
+------+---------+------------+-----------------+---------+------------+-------------
-+----------+--------------------+
|   234|        0|           5|                2|        5|        null|
3|         0|                   3|
```

In [6]:
```
print('每个特征包含的种类数：')
print(['特征' + str(c) + '的个数' + str(user_profile_df.groupBy(c).count().count()) f
print('表中总共有%d行'%user_profile_df.count())
print('每列各自的行数：')
[str(c) + '列的行数：' + str(user_profile_df.select(c).dropna().count()) for c in use
```

```
每个特征包含的种类数：
['特征cms_segid的个数97', '特征cms_group_id的个数13', '特征final_gender_code的个数2',
'特征age_level的个数7', '特征pvalue_level的个数4', '特征shopping_level的个数3', '特征oc
cupation的个数2', '特征new_user_class_level的个数5']
表中总共有1061768行
每列各自的行数：
```

Out[6]:
```
['userId列的行数：1061768',
 'cms_segid列的行数：1061768',
 'cms_group_id列的行数：1061768',
 'final_gender_code列的行数：1061768',
 'age_level列的行数：1061768',
 'pvalue_level列的行数：485851',
 'shopping_level列的行数：1061768',
 'occupation列的行数：1061768',
 'new_user_class_level列的行数：716848']
```

In [7]:
```
user_profile_df.groupBy('final_gender_code').count().show()
```

```
+-----------------+------+
|final_gender_code| count|
+-----------------+------+
|                1|377517|
|                2|684251|
+-----------------+------+
```

1. 处理缺失值，两列数据有空值

In [8]:
```
# 有缺失值的特征
user_profile_df.groupBy('pvalue_level').count().show()
user_profile_df.groupBy('new_user_class_level').count().show()
```

```
+------------+------+
|pvalue_level| count|
+------------+------+
|        null|575917|
|           1|154436|
|           3| 37759|
|           2|293656|
+------------+------+

+--------------------+------+
|new_user_class_level| count|
+--------------------+------+
|                null|344920|
|                   1| 80548|
|                   3|173047|
|                   4|138833|
|                   2|324420|
+--------------------+------+
```

1.1 先不删除确实严重的特征

缺失值数据处理

注意，一般情况下：

- 缺失率低于10%：可直接进行相应的填充，如默认值、均值、算法拟合等等；
- 高于10%：往往会考虑舍弃该特征
- 特征处理，如1维转多维

但根据我们的经验，我们的广告推荐其实和用户的消费水平、用户所在城市等级都有比较大的关联，因此在这里pvalue_level、new_user_class_level都是比较重要的特征，我们不考虑舍弃

缺失值处理方案：

1. 填充方案：结合用户的其他特征值，利用随机森林算法进行预测；但产生了大量人为构建的数据，一定程度上增加了数据的噪音
2. 把变量映射到高维空间：如pvalue_level的1维数据，转换成是否1、是否2、是否3、是否缺失的4维数据；这样保证了所有原始数据不变，同时能提高精确度，但这样会导致数据变得比较稀疏，如果样本量很小，反而会导致样本效果较差，因此也不能滥用

1.1.1 填充方案之 利用随机森林对缺失值进行预测

a. 利用随机森林对pvalue_level的缺失值进行预测

In [10]:
```
# 随机森林填充缺失值
# 随机森林模型的输入值类型必须是 LabeledPoint类型
from pyspark.mllib.regression import LabeledPoint
# dropna()整行删除，某行有一个空值 就删掉整行
# mlib是基于rdd的，因此后续处理过程需要转化成rdd
train_data = user_profile_df.dropna(subset=['pvalue_level']).rdd.\
    map(lambda r:LabeledPoint(r.pvalue_level-1,\
    # final_gender_code原数据中是1、2，随机森林模型使用时要变成0，1
    [r.cms_segid, r.cms_group_id, r.final_gender_code-1, r.age_level, r.shopping_level
train_data.collect()
```

Out[10]:
```
[LabeledPoint(0.0, [5.0,2.0,1.0,2.0,2.0,1.0]),
 LabeledPoint(1.0, [0.0,8.0,0.0,2.0,2.0,0.0]),
 LabeledPoint(1.0, [49.0,6.0,1.0,6.0,2.0,0.0]),
 LabeledPoint(1.0, [44.0,5.0,1.0,5.0,2.0,0.0]),
 LabeledPoint(0.0, [2.0,1.0,1.0,1.0,2.0,0.0]),
 LabeledPoint(1.0, [43.0,5.0,1.0,5.0,2.0,0.0]),
 LabeledPoint(1.0, [5.0,2.0,1.0,2.0,2.0,1.0]),
 LabeledPoint(1.0, [55.0,8.0,0.0,2.0,1.0,0.0]),
 LabeledPoint(1.0, [33.0,4.0,1.0,4.0,2.0,0.0]),
 LabeledPoint(1.0, [0.0,4.0,1.0,4.0,2.0,0.0]),
 LabeledPoint(1.0, [0.0,4.0,1.0,4.0,2.0,0.0]),
 LabeledPoint(0.0, [8.0,2.0,1.0,2.0,2.0,0.0]),
 LabeledPoint(0.0, [31.0,4.0,1.0,4.0,2.0,0.0]),
 LabeledPoint(1.0, [20.0,3.0,1.0,3.0,2.0,0.0]),
 LabeledPoint(1.0, [33.0,4.0,1.0,4.0,2.0,0.0]),
 LabeledPoint(1.0, [36.0,5.0,1.0,5.0,0.0,0.0]),
 LabeledPoint(1.0, [20.0,3.0,1.0,3.0,2.0,0.0]),
 LabeledPoint(1.0, [8.0,2.0,1.0,2.0,2.0,0.0]),
 LabeledPoint(1.0, [0.0,4.0,1.0,4.0,2.0,0.0]),
```

In [11]:
```
# # 延申学习：LabeledPoint 与 稀疏矩阵SparseVector
# from pyspark.mllib.linalg import SparseVector
# from pyspark.mllib.regression import LabeledPoint

# # Create a labeled point with a positive label and a dense feature vector.
# pos = LabeledPoint(1.0, [1.0, 0.0, 3.0])

# # Create a labeled point with a negative label and a sparse feature vector.
# neg = LabeledPoint(0.0, SparseVector(3, [0, 2], [1.0, 3.0]))
```

In [12]:
```
from pyspark.mllib.tree import RandomForest
# 训练分类模型
# 参数1 训练的数据
#参数2 目标值的分类个数 0,1,2
#参数3 特征中是否包含分类的特征 {2:2,3:7} {2:2} 表示 在特征中 第三个特征是分类的: 有两
#参数4 随机森林中 树的棵数
model = RandomForest.trainClassifier(train_data,3,{2:2, 3:7, 4:3, 5:2},5)
```

In [13]:
```
# 看看上上上个cell中哪个特征是分类特征
# user_profile_df.groupBy('cms_segid').count().show()# 种类太多
# user_profile_df.groupBy('cms_group_id').count().show()# 种类太多
# user_profile_df.groupBy('final_gender_code').count().show()# 1、2 ->0、1
# user_profile_df.groupBy('age_level').count().show() # 0-6
# user_profile_df.groupBy('shopping_level').count().show() # 1、2、3 -> 0、1、2
# user_profile_df.groupBy('occupation').count().show() # 0、1
```

In [14]:
```
# 取出pvalue_level缺失的那些行
pl_na_df = user_profile_df.fillna(-1).where('pvalue_level=-1')
```

In [15]:
```
# 随机森林模型 使用的是rdd，准备输入模型的rdd
rdd = pl_na_df.rdd.map(lambda r:(r.cms_segid, r.cms_group_id, r.final_gender_code-1,
predicts = model.predict(rdd)
print(predicts.take(20))
print('预测值总数',predicts.count())
# 这里注意predict参数，如果是预测多个，那么参数必须是直接有列表构成的rdd参数，而不能是
# 因此这里经过map函数处理，将每一行数据转换为普通的列表数据
```

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0,
1.0, 1.0, 1.0]
预测值总数 575917
```

In [16]:
```
# 这里数据量比较小，直接转换为pandas dataframe来处理，因为方便，但注意如果数据量较大不
temp = predicts.map(lambda x:int(x)).collect()
pdf = pl_na_df.toPandas()
import numpy as np
# 在pandas df的基础上直接替换掉列数据
pdf["pvalue_level"] = np.array(temp) + 1  # 注意+1 还原预测值
pdf

```

Out[16]:

	userId	cms_segid	cms_group_id	final_gender_code	age_level	pvalue_level	shopping_l
0	234	0	5	2	5	2	
1	1670	0	4	2	4	2	
2	2545	0	10	1	4	2	
3	6211	0	9	1	3	2	
4	9293	0	5	2	5	2	
...	...	...	...	...	...	...	
575912	1137329	0	4	2	4	2	
575913	1137582	0	9	1	3	2	
575914	1137955	0	3	2	3	1	
575915	1138545	0	4	2	4	2	
575916	1139632	0	7	1	1	2	

575917 rows × 9 columns

In [17]:
```
# 与非缺失数据进行拼接，完成pvalue_level的缺失值预测
new_user_profile_df = user_profile_df.dropna(subset=["pvalue_level"]).unionAll(spark.
new_user_profile_df.show()
# 注意：unionAll的使用，两个df的表结构必须完全一样
```

...

b. 利用随机森林对new_user_class_level的缺失值进行预测

In [18]:
```
from pyspark.mllib.regression import LabeledPoint

# 选出new_user_class_level全部的
train_data2 = user_profile_df.dropna(subset=["new_user_class_level"]).rdd.map(
    lambda r:LabeledPoint(r.new_user_class_level - 1,\
                          [r.cms_segid, r.cms_group_id, r.final_gender_code-1, r.age_
)
from pyspark.mllib.tree import RandomForest
model2 = RandomForest.trainClassifier(train_data2, 4, {2:2, 3:7, 4:3, 5:2}, 5)
```

In [19]:
```
nul_na_df = user_profile_df.na.fill(-1).where("new_user_class_level=-1")
nul_na_df.show(10)

def row(r):
    return r.cms_segid, r.cms_group_id, r.final_gender_code, r.age_level, r.shopping_

rdd2 = nul_na_df.rdd.map(row)
predicts2 = model2.predict(rdd2)
predicts2.take(20)
```

```
+------+---------+-----------+-----------------+---------+-----------+-------------
-+----------+--------------------+
|userId|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_leve
l|occupation|new_user_class_level|
+------+---------+-----------+-----------------+---------+-----------+-------------
-+----------+--------------------+
|   612|        0|          8|                1|        2|          2|
3|         0|                  -1|
|  1670|        0|          4|                2|        4|         -1|
1|         0|                  -1|
|  2545|        0|         10|                1|        4|         -1|
3|         0|                  -1|
| 10549|        0|          4|                2|        4|          2|
3|         0|                  -1|
| 10812|        0|          4|                2|        4|         -1|
2|         0|                  -1|
| 10912|        0|          4|                2|        4|          2|
3|         0|                  -1|
| 12620|        0|          4|                2|        4|         -1|
```

总结：可以发现由于这两个字段的缺失过多，所以预测出来的值已经大大失真，但如果缺失率在10%以下，这种方法是比较有效的一种

1.1.2 低维转高维方式

1.1.2. 低维转高维方式

因此我们接下来采用将变量映射到高维空间的方法来处理数据，即将缺失项也当做一个单独的特征来对待，保证数据的原始性

由于该思想正好和热独编码实现方法一样，因此这里直接使用热独编码方式处理数据

In [23]:

```
from pyspark.ml.feature import OneHotEncoder
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline
from pyspark.sql.types import StringType
user_profile_df = user_profile_df.fillna(-1)
user_profile_df.show()
# 热独编码时，必须先将待处理字段 转为 字符串类型才可处理
user_profile_df = user_profile_df.withColumn('pvalue_level',user_profile_df.pvalue_lev
    .withColumn('new_user_class_level',user_profile_df.new_user_class_level.cast(Strin
user_profile_df.printSchema()
# 对pvalue_level进行热独编码，求值
stringindexer = StringIndexer(inputCol='pvalue_level',outputCol='pl_onehot_feature')
encoder = OneHotEncoder(dropLast=False,inputCol='pl_onehot_feature',outputCol='pl_on
pipeline = Pipeline(stages=[stringindexer,encoder])
pipeline_fit = pipeline.fit(user_profile_df)
user_profile_df2 = pipeline_fit.transform(user_profile_df)
# pl_onehot_value列的值为稀疏向量，存储热独编码的结果
user_profile_df2.printSchema()
user_profile_df2.show()
```

```
+------+--------+-----------+----------------+---------+-----------+-------------
-+---------+-------------------+
|userId|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_leve
l|occupation|new_user_class_level|
+------+--------+-----------+----------------+---------+-----------+-------------
-+---------+-------------------+
|   234|       0|          5|               2|        5|          -1|
3|         0|                  3|
|   523|       5|          2|               2|        2|           1|
3|         1|                  2|
|   612|       0|          8|               1|        2|           2|
3|         0|                 -1|
|  1670|       0|          4|               2|        4|          -1|
1|         0|                 -1|
|  2545|       0|         10|               1|        4|          -1|
3|         0|                 -1|
|  3644|      49|          6|               2|        6|           2|
3|         0|                  2|
|  5777|      44|          5|               2|        5|           2|
3|         0|                  2|
|  6211|       0|          9|               1|        3|          -1|
3|         0|                  2|
|  6355|       2|          1|               2|        1|           1|
3|         0|                  4|
|  6823|      43|          5|               2|        5|           2|
3|         0|                  1|
|  6972|       5|          2|               2|        2|           2|
3|         1|                  2|
|  9293|       0|          5|               2|        5|          -1|
3|         0|                  4|
|  9510|      55|          8|               1|        2|           2|
2|         0|                  2|
| 10122|      33|          4|               2|        4|           2|
3|         0|                  2|
| 10549|       0|          4|               2|        4|           2|
3|         0|                 -1|
| 10812|       0|          4|               2|        4|          -1|
```

```
2|         0|                 -1|
| 10912|       0|          4|                2|        4|           2|
3|         0|                 -1|
| 10996|       0|          5|                2|        5|          -1|
3|         0|                  4|
| 11256|       8|          2|                2|        2|           1|
3|         0|                  3|
| 11310|      31|          4|                2|        4|           1|
3|         0|                  4|
+------+---------+------------+-----------------+---------+------------+-------------
-+----------+-------------------+
only showing top 20 rows

root
 |-- userId: integer (nullable = true)
 |-- cms_segid: integer (nullable = true)
 |-- cms_group_id: integer (nullable = true)
 |-- final_gender_code: integer (nullable = true)
 |-- age_level: integer (nullable = true)
 |-- pvalue_level: string (nullable = true)
 |-- shopping_level: integer (nullable = true)
 |-- occupation: integer (nullable = true)
 |-- new_user_class_level: string (nullable = true)

root
 |-- userId: integer (nullable = true)
 |-- cms_segid: integer (nullable = true)
 |-- cms_group_id: integer (nullable = true)
 |-- final_gender_code: integer (nullable = true)
 |-- age_level: integer (nullable = true)
 |-- pvalue_level: string (nullable = true)
 |-- shopping_level: integer (nullable = true)
 |-- occupation: integer (nullable = true)
 |-- new_user_class_level: string (nullable = true)
 |-- pl_onehot_feature: double (nullable = true)
 |-- pl_onehot_value: vector (nullable = true)

+------+---------+------------+-----------------+---------+------------+-------------
-+----------+-------------------+-----------------+---------------+
|userId|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_leve
l|occupation|new_user_class_level|pl_onehot_feature|pl_onehot_value|
+------+---------+------------+-----------------+---------+------------+-------------
-+----------+-------------------+-----------------+---------------+
|   234|       0|          5|                2|        5|          -1|
3|         0|                  3|              0.0|  (4,[0],[1.0])|
|   523|       5|          2|                2|        2|           1|
3|         1|                  2|              2.0|  (4,[2],[1.0])|
|   612|       0|          8|                1|        2|           2|
3|         0|                 -1|              1.0|  (4,[1],[1.0])|
|  1670|       0|          4|                2|        4|          -1|
1|         0|                 -1|              0.0|  (4,[0],[1.0])|
|  2545|       0|         10|                1|        4|          -1|
3|         0|                 -1|              0.0|  (4,[0],[1.0])|
|  3644|      49|          6|                2|        6|           2|
3|         0|                  2|              1.0|  (4,[1],[1.0])|
|  5777|      44|          5|                2|        5|           2|
3|         0|                  2|              1.0|  (4,[1],[1.0])|
```

```
|  6211|        0|          9|                1|         3|          -1|
3|         0|                  2|              0.0|  (4,[0],[1.0])|
|  6355|        2|          1|                2|         1|           1|
3|         0|                  4|              2.0|  (4,[2],[1.0])|
|  6823|       43|          5|                2|         5|           2|
3|         0|                  1|              1.0|  (4,[1],[1.0])|
|  6972|        5|          2|                2|         2|           2|
3|         1|                  2|              1.0|  (4,[1],[1.0])|
|  9293|        0|          5|                2|         5|          -1|
3|         0|                  4|              0.0|  (4,[0],[1.0])|
|  9510|       55|          8|                1|         2|           2|
2|         0|                  2|              1.0|  (4,[1],[1.0])|
| 10122|       33|          4|                2|         4|           2|
3|         0|                  2|              1.0|  (4,[1],[1.0])|
| 10549|        0|          4|                2|         4|           2|
3|         0|                 -1|              1.0|  (4,[1],[1.0])|
| 10812|        0|          4|                2|         4|          -1|
2|         0|                 -1|              0.0|  (4,[0],[1.0])|
| 10912|        0|          4|                2|         4|           2|
3|         0|                 -1|              1.0|  (4,[1],[1.0])|
| 10996|        0|          5|                2|         5|          -1|
3|         0|                  4|              0.0|  (4,[0],[1.0])|
| 11256|        8|          2|                2|         2|           1|
3|         0|                  3|              2.0|  (4,[2],[1.0])|
| 11310|       31|          4|                2|         4|           1|
3|         0|                  4|              2.0|  (4,[2],[1.0])|
+------+---------+-----------+-----------------+---------+-----------+-------------
-+----------+-------------------+-----------------+--------------+
only showing top 20 rows
```

In [29]:
```
# 使用热编码转换new_user_class_level的一维数据为多维
stringindexer = StringIndexer(inputCol='new_user_class_level',outputCol='nucl_onehot_
encoder = OneHotEncoder(dropLast=False,inputCol='nucl_onehot_feature',outputCol='nuc
pipeline = Pipeline(stages=[stringindexer,encoder])
pipeline_fit = pipeline.fit(user_profile_df2)
user_profile_df3 = pipeline_fit.transform(user_profile_df2)
user_profile_df3.printSchema()
user_profile_df3.show()
```

```
root
 |-- userId: integer (nullable = true)
 |-- cms_segid: integer (nullable = true)
 |-- cms_group_id: integer (nullable = true)
 |-- final_gender_code: integer (nullable = true)
 |-- age_level: integer (nullable = true)
 |-- pvalue_level: string (nullable = true)
 |-- shopping_level: integer (nullable = true)
 |-- occupation: integer (nullable = true)
 |-- new_user_class_level: string (nullable = true)
 |-- pl_onehot_feature: double (nullable = true)
 |-- pl_onehot_value: vector (nullable = true)
 |-- nucl_onehot_feature: double (nullable = true)
 |-- nucl_onehot_value: vector (nullable = true)

+------+---------+-----------+-----------------+---------+------------+-------------
-+----------+--------------------+-----------------+---------------+-----------------
--+-----------------+
|userId|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_leve
```

In [30]:
```
# 如何将用户特征合并?
#例如我们要合并上表中的三列为一列："age_level", "pl_onehot_value", "nucl_onehot_value"
from pyspark.ml.feature import VectorAssembler
feature_df = VectorAssembler().setInputCols(["age_level", "pl_onehot_value", "nucl_one
    .transform(user_profile_df3)
feature_df.show()
```

```
+------+--------+----------+---------------+---------+----------+-------------
-+---------+------------------+----------------+--------------+----------------
--+----------------+------------------+
|userId|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_leve
l|occupation|new_user_class_level|pl_onehot_feature|pl_onehot_value|nucl_onehot_featu
re|nucl_onehot_value|            features|
+------+--------+----------+---------------+---------+----------+-------------
-+---------+------------------+----------------+--------------+----------------
--+----------------+------------------+
|   234|       0|         5|              2|        5|        -1|
3|         0|                 3|             0.0|  (4,[0],[1.0])|
2.0|    (5,[2],[1.0])|(10,[0,1,7],[5.0,...|
|   523|       5|         2|              2|        2|         1|
3|         1|                 2|             2.0|  (4,[2],[1.0])|
1.0|    (5,[1],[1.0])|(10,[0,3,6],[2.0,...|
|   612|       0|         8|              1|        2|         2|
3|         0|                -1|             1.0|  (4,[1],[1.0])|
0.0|    (5,[0],[1.0])|(10,[0,2,5],[2.0,...|
|  1670|       0|         4|              2|        4|        -1|
1|         0|                -1|             0.0|  (4,[0],[1.0])|
0.0|    (5,[0],[1.0])|(10,[0,1,5],[4.0,...|
|  2545|       0|        10|              1|        4|        -1|
3|         0|                -1|             0.0|  (4,[0],[1.0])|
0.0|    (5,[0],[1.0])|(10,[0,1,5],[4.0,...|
|  3644|      49|         6|              2|        6|         2|
3|         0|                 2|             1.0|  (4,[1],[1.0])|
1.0|    (5,[1],[1.0])|(10,[0,2,6],[6.0,...|
|  5777|      44|         5|              2|        5|         2|
3|         0|                 2|             1.0|  (4,[1],[1.0])|
1.0|    (5,[1],[1.0])|(10,[0,2,6],[5.0,...|
|  6211|       0|         9|              1|        3|        -1|
3|         0|                 2|             0.0|  (4,[0],[1.0])|
1.0|    (5,[1],[1.0])|(10,[0,1,6],[3.0,...|
|  6355|       2|         1|              2|        1|         1|
3|         0|                 4|             2.0|  (4,[2],[1.0])|
3.0|    (5,[3],[1.0])|(10,[0,3,8],[1.0,...|
|  6823|      43|         5|              2|        5|         2|
3|         0|                 1|             1.0|  (4,[1],[1.0])|
4.0|    (5,[4],[1.0])|(10,[0,2,9],[5.0,...|
|  6972|       5|         2|              2|        2|         2|
3|         1|                 2|             1.0|  (4,[1],[1.0])|
1.0|    (5,[1],[1.0])|(10,[0,2,6],[2.0,...|
|  9293|       0|         5|              2|        5|        -1|
3|         0|                 4|             0.0|  (4,[0],[1.0])|
3.0|    (5,[3],[1.0])|(10,[0,1,8],[5.0,...|
|  9510|      55|         8|              1|        2|         2|
2|         0|                 2|             1.0|  (4,[1],[1.0])|
1.0|    (5,[1],[1.0])|(10,[0,2,6],[2.0,...|
| 10122|      33|         4|              2|        4|         2|
3|         0|                 2|             1.0|  (4,[1],[1.0])|
```

```
1.0|     (5,[1],[1.0])|(10,[0,2,6],[4.0,...|
| 10549|        0|           4|                2|        4|           2|
3|         0|                  -1|              1.0|   (4,[1],[1.0])|
0.0|     (5,[0],[1.0])|(10,[0,2,5],[4.0,...|
| 10812|        0|           4|                2|        4|          -1|
2|         0|                  -1|              0.0|   (4,[0],[1.0])|
0.0|     (5,[0],[1.0])|(10,[0,1,5],[4.0,...|
| 10912|        0|           4|                2|        4|           2|
3|         0|                  -1|              1.0|   (4,[1],[1.0])|
0.0|     (5,[0],[1.0])|(10,[0,2,5],[4.0,...|
| 10996|        0|           5|                2|        5|          -1|
3|         0|                   4|              0.0|   (4,[0],[1.0])|
3.0|     (5,[3],[1.0])|(10,[0,1,8],[5.0,...|
| 11256|        8|           2|                2|        2|           1|
3|         0|                   3|              2.0|   (4,[2],[1.0])|
2.0|     (5,[2],[1.0])|(10,[0,3,7],[2.0,...|
| 11310|       31|           4|                2|        4|           1|
3|         0|                   4|              2.0|   (4,[2],[1.0])|
3.0|     (5,[3],[1.0])|(10,[0,3,8],[4.0,...|
+------+--------+------------+-----------------+---------+------------+-------------
-+----------+-------------------+-----------------+--------------+-----------------
--+-----------------+-------------------+
only showing top 20 rows
```

In [31]:
```
1 feature_df.select('features').show()
```

```
+--------------------+
|            features|
+--------------------+
|(10,[0,1,7],[5.0,...|
|(10,[0,3,6],[2.0,...|
|(10,[0,2,5],[2.0,...|
|(10,[0,1,5],[4.0,...|
|(10,[0,1,5],[4.0,...|
|(10,[0,2,6],[6.0,...|
|(10,[0,2,6],[5.0,...|
|(10,[0,1,6],[3.0,...|
|(10,[0,3,8],[1.0,...|
|(10,[0,2,9],[5.0,...|
|(10,[0,2,6],[2.0,...|
|(10,[0,1,8],[5.0,...|
|(10,[0,2,6],[2.0,...|
|(10,[0,2,6],[4.0,...|
|(10,[0,2,5],[4.0,...|
|(10,[0,1,5],[4.0,...|
|(10,[0,2,5],[4.0,...|
```

In [ ]:

```
# 我该选取那些特征呢？
# 除了前面处理的pvalue_level和new_user_class_level需要作为特征以外，(能体现出用户的购
# 分类特征值：个数
# - cms_segid:  97
# - cms_group_id:  13
# - final_gender_code:  2
# - age_level:  7
# - shopping_level:  3
# - occupation:  2
# - pvalue_level
# - new_user_class_level
# - price
# 根据经验，以上几个分类特征都一定程度能体现用户在购物方面的特征，且类别都较少，都可以
```