基于LR的点击率预测模型训练

本小节主要根据广告点击样本数据集(raw_sample)、广告基本特征数据集(ad_feature)、用户基本信息数据集(user_profile)构建出了一个完整的样本数据集，并按日期划分为了训练集(前七天)和测试集(最后一天)，利用逻辑回归进行训练。

训练模型时，通过对类别特征数据进行处理，一定程度达到提高了模型的效果

分析对比两种模型：

1. 训练CTRModel_Normal：直接将对应的特征的特征值组合成对应的特征向量进行训练

In [ ]:
```
'''
# 剔除冗余、不需要的字段
useful_cols = [
    #
    # 时间字段，划分训练集和测试集
    "timestamp",
    # label目标值字段
    "clk",
    # 特征值字段
    "pid_value",        # 资源位的特征向量
    "price",    # 广告价格
    "cms_segid",    # 用户微群ID
    "cms_group_id",    # 用户组ID
    "final_gender_code",    # 用户性别特征，[1,2]
    "age_level",    # 年龄等级，1-
    "shopping_level",
    "occupation",
    "pl_onehot_value",
    "nucl_onehot_value"
]
# 筛选指定字段数据，构建新的数据集
datasets_1 = datasets.select(*useful_cols)
'''
```

2. 训练CTRModel_AllOneHot

- "pid_value", 类别型特征，已被转换为多维特征==> 2维
- "price", 统计型特征 ===> 1维
- "cms_segid", 类别型特征，约97个分类 ===> 1维
- "cms_group_id", 类别型特征，约13个分类 ==> 1维
- "final_gender_code", 类别型特征，2个分类 ==> 1维
- "age_level", 类别型特征，7个分类 ==> 1维
- "shopping_level", 类别型特征，3个分类 ==> 1维
- "occupation", 类别型特征，2个分类 ==> 1维
- "pl_onehot_value", 类别型特征，已被转换为多维特征 ==> 4维
- "nucl_onehot_value" 类别型特征，已被转换为多维特征 ==> 5维

类别性特征都可以考虑进行热独编码，将单一变量变为多变量，相当于增加了相关特征的数量

- "cms_segid", 类别型特征，约97个分类 ===> 97维 舍弃
- "cms_group_id", 类别型特征，约13个分类 ==> 13维
- "final_gender_code", 类别型特征，2个分类 ==> 2维
- "age_level", 类别型特征，7个分类 ==>7维
- "shopping_level", 类别型特征，3个分类 ==> 3维
- "occupation", 类别型特征，2个分类 ==> 2维

但由于cms_segid分类过多，这里考虑舍弃，避免数据过于稀疏

结论:

对比前面的result_1的预测结果，能发现这里的预测率稍微准确了一点，这里top20里出现了3个点击的，但前面的只出现了1个

因此可见对特征的细化处理，已经帮助我们提高模型的效果的

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
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 = 'createCTRModelByLR'
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')
#     ("spark.executor.instances", 1)    # 设置spark executor数量，yarn时起作用
)
conf.setAll(config)

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

In [4]:
```
'''
raw_sample
      pid
ad_feature
      price
user_profile
    - 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
'''
```

1.raw_sample - pid

In [6]:
```
_raw_sample_df1 = spark.read.csv('/data/raw_sample.csv',header=True)
_raw_sample_df1.show()
_raw_sample_df1.printSchema()
```

```
+------+----------+----------+-----------+------+---+
|  user|time_stamp|adgroup_id|        pid|nonclk|clk|
+------+----------+----------+-----------+------+---+
|581738|1494137644|         1|430548_1007|     1|  0|
|449818|1494638778|         3|430548_1007|     1|  0|
|914836|1494650879|         4|430548_1007|     1|  0|
|914836|1494651029|         5|430548_1007|     1|  0|
|399907|1494302958|         8|430548_1007|     1|  0|
|628137|1494524935|         9|430548_1007|     1|  0|
|298139|1494462593|         9|430539_1007|     1|  0|
|775475|1494561036|         9|430548_1007|     1|  0|
|555266|1494307136|        11|430539_1007|     1|  0|
|117840|1494036743|        11|430548_1007|     1|  0|
|739815|1494115387|        11|430539_1007|     1|  0|
|623911|1494625301|        11|430548_1007|     1|  0|
|623911|1494451608|        11|430548_1007|     1|  0|
|421590|1494034144|        11|430548_1007|     1|  0|
|976358|1494156949|        13|430548_1007|     1|  0|
|286630|1494218579|        13|430539_1007|     1|  0|
|286630|1494289247|        13|430539_1007|     1|  0|
|771431|1494153867|        13|430548_1007|     1|  0|
|707120|1494220810|        13|430548_1007|     1|  0|
|530454|1494293746|        13|430548_1007|     1|  0|
+------+----------+----------+-----------+------+---+
only showing top 20 rows

root
 |-- user: string (nullable = true)
 |-- time_stamp: string (nullable = true)
 |-- adgroup_id: string (nullable = true)
 |-- pid: string (nullable = true)
 |-- nonclk: string (nullable = true)
 |-- clk: string (nullable = true)
```

In [7]:
```
from pyspark.sql.types import StringType, StructField, IntegerType, FloatType, LongT
_raw_sample_df2 = _raw_sample_df1.withColumn('user',_raw_sample_df1.user.cast(Integer
    withColumn('time_stamp',_raw_sample_df1.time_stamp.cast(LongType())).withColumnRe
    withColumn("adgroup_id", _raw_sample_df1.adgroup_id.cast(IntegerType())).withColu
    withColumn("pid", _raw_sample_df1.pid.cast(StringType())).\
    withColumn("nonclk", _raw_sample_df1.nonclk.cast(IntegerType())).\
    withColumn("clk", _raw_sample_df1.clk.cast(IntegerType()))
_raw_sample_df2.printSchema()
_raw_sample_df2.show()
```

```
root
 |-- userId: integer (nullable = true)
 |-- timestamp: long (nullable = true)
 |-- adgroupId: integer (nullable = true)
 |-- pid: string (nullable = true)
 |-- nonclk: integer (nullable = true)
 |-- clk: integer (nullable = true)

+------+----------+---------+-----------+------+---+
|userId| timestamp|adgroupId|        pid|nonclk|clk|
+------+----------+---------+-----------+------+---+
|581738|1494137644|        1|430548_1007|     1|  0|
|449818|1494638778|        3|430548_1007|     1|  0|
|914836|1494650879|        4|430548_1007|     1|  0|
|914836|1494651029|        5|430548_1007|     1|  0|
|399907|1494302958|        8|430548_1007|     1|  0|
|628137|1494524935|        9|430548_1007|     1|  0|
|298139|1494462593|        9|430539_1007|     1|  0|
|775475|1494561036|        9|430548_1007|     1|  0|
|555266|1494307136|       11|430539_1007|     1|  0|
|117840|1494036743|       11|430548_1007|     1|  0|
|739815|1494115387|       11|430539_1007|     1|  0|
|623911|1494625301|       11|430548_1007|     1|  0|
|623911|1494451608|       11|430548_1007|     1|  0|
|421590|1494034144|       11|430548_1007|     1|  0|
|976358|1494156949|       13|430548_1007|     1|  0|
|286630|1494218579|       13|430539_1007|     1|  0|
|286630|1494289247|       13|430539_1007|     1|  0|
|771431|1494153867|       13|430548_1007|     1|  0|
|707120|1494220810|       13|430548_1007|     1|  0|
|530454|1494293746|       13|430548_1007|     1|  0|
+------+----------+---------+-----------+------+---+
only showing top 20 rows
```

In [8]:
```
from pyspark.ml.feature import OneHotEncoder
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline
stringindexer = StringIndexer(inputCol='pid',outputCol='pid_feature')
encoder = OneHotEncoder(dropLast=False,inputCol='pid_feature',outputCol='pid_value')
pipeline = Pipeline(stages=[stringindexer,encoder])
pipeline_fit= pipeline.fit(_raw_sample_df2)
raw_sample_df = pipeline_fit.transform(_raw_sample_df2)
raw_sample_df.show()

'''pid和特征的对应关系
430548_1007：0
430549_1007：1
'''
```

```
+------+----------+---------+----------+------+---+-----------+-------------+
|userId| timestamp|adgroupId|       pid|nonclk|clk|pid_feature|    pid_value|
+------+----------+---------+----------+------+---+-----------+-------------+
|581738|1494137644|        1|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|449818|1494638778|        3|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|914836|1494650879|        4|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|914836|1494651029|        5|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|399907|1494302958|        8|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|628137|1494524935|        9|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|298139|1494462593|        9|430539_1007|     1|  0|        1.0|(2,[1],[1.0])|
|775475|1494561036|        9|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|555266|1494307136|       11|430539_1007|     1|  0|        1.0|(2,[1],[1.0])|
|117840|1494036743|       11|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|739815|1494115387|       11|430539_1007|     1|  0|        1.0|(2,[1],[1.0])|
|623911|1494625301|       11|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|623911|1494451608|       11|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|421590|1494034144|       11|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|976358|1494156949|       13|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|286630|1494218579|       13|430539_1007|     1|  0|        1.0|(2,[1],[1.0])|
|286630|1494289247|       13|430539_1007|     1|  0|        1.0|(2,[1],[1.0])|
|771431|1494153867|       13|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|707120|1494220810|       13|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
|530454|1494293746|       13|430548_1007|     1|  0|        0.0|(2,[0],[1.0])|
+------+----------+---------+----------+------+---+-----------+-------------+
only showing top 20 rows
```

Out[8]: 'pid和特征的对应关系\n430548_1007：0\n430549_1007：1\n'

2.广告基本信息ad_feature - price

In [10]:
```
_ad_feature_df = spark.read.csv('/data/ad_feature.csv',header=True)
_ad_feature_df.printSchema()
_ad_feature_df.show()
```

```
root
 |-- adgroup_id: string (nullable = true)
 |-- cate_id: string (nullable = true)
 |-- campaign_id: string (nullable = true)
 |-- customer: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- price: string (nullable = true)

+----------+-------+-----------+--------+------+-----+
|adgroup_id|cate_id|campaign_id|customer| brand|price|
+----------+-------+-----------+--------+------+-----+
|     63133|   6406|      83237|       1| 95471|170.0|
|    313401|   6406|      83237|       1| 87331|199.0|
|    248909|    392|      83237|       1| 32233| 38.0|
|    208458|    392|      83237|       1|174374|139.0|
|    110847|   7211|     135256|       2|145952|32.99|
|    607788|   6261|     387991|       6|207800|199.0|
|    375706|   4520|     387991|       6|  NULL| 99.0|
|     11115|   7213|     139747|       9|186847| 33.0|
|     24484|   7207|     139744|       9|186847| 19.0|
|     28589|   5953|     395195|      13|  NULL|428.0|
|     23236|   5953|     395195|      13|  NULL|368.0|
|    300556|   5953|     395195|      13|  NULL|639.0|
|     92560|   5953|     395195|      13|  NULL|368.0|
|    590965|   4284|      28145|      14|454237|249.0|
|    529913|   4284|      70206|      14|  NULL|249.0|
|    546930|   4284|      28145|      14|  NULL|249.0|
|    639794|   6261|      70206|      14| 37004| 89.9|
|    335413|   4284|      28145|      14|  NULL|249.0|
|    794890|   4284|      70206|      14|454237|249.0|
|    684020|   6261|      70206|      14| 37004| 99.0|
+----------+-------+-----------+--------+------+-----+
only showing top 20 rows
```

In [11]:
```
from pyspark.sql.types import IntegerType, FloatType
ad_feature_df = _ad_feature_df.\
    withColumn("adgroup_id", _ad_feature_df.adgroup_id.cast(IntegerType())).withColumn
    withColumn("cate_id", _ad_feature_df.cate_id.cast(IntegerType())).withColumnRename
    withColumn("campaign_id", _ad_feature_df.campaign_id.cast(IntegerType())).withColu
    withColumn("customer", _ad_feature_df.customer.cast(IntegerType())).withColumnRena
    withColumn("brand", _ad_feature_df.brand.cast(IntegerType())).withColumnRenamed("b
    withColumn("price", _ad_feature_df.price.cast(FloatType()))
ad_feature_df.printSchema()
ad_feature_df.show()
```

```
root
 |-- adgroupId: integer (nullable = true)
 |-- cateId: integer (nullable = true)
 |-- campaignId: integer (nullable = true)
 |-- customerId: integer (nullable = true)
 |-- brandId: integer (nullable = true)
 |-- price: float (nullable = true)

+---------+------+----------+----------+-------+-----+
|adgroupId|cateId|campaignId|customerId|brandId|price|
+---------+------+----------+----------+-------+-----+
|    63133|  6406|     83237|         1|  95471|170.0|
|   313401|  6406|     83237|         1|  87331|199.0|
|   248909|   392|     83237|         1|  32233| 38.0|
|   208458|   392|     83237|         1| 174374|139.0|
|   110847|  7211|    135256|         2| 145952|32.99|
|   607788|  6261|    387991|         6| 207800|199.0|
|   375706|  4520|    387991|         6|   null| 99.0|
|    11115|  7213|    139747|         9| 186847| 33.0|
|    24484|  7207|    139744|         9| 186847| 19.0|
|    28589|  5953|    395195|        13|   null|428.0|
|    23236|  5953|    395195|        13|   null|368.0|
|   300556|  5953|    395195|        13|   null|639.0|
|    92560|  5953|    395195|        13|   null|368.0|
|   590965|  4284|     28145|        14| 454237|249.0|
|   529913|  4284|     70206|        14|   null|249.0|
|   546930|  4284|     28145|        14|   null|249.0|
|   639794|  6261|     70206|        14|  37004| 89.9|
|   335413|  4284|     28145|        14|   null|249.0|
|   794890|  4284|     70206|        14| 454237|249.0|
|   684020|  6261|     70206|        14|  37004| 99.0|
+---------+------+----------+----------+-------+-----+
only showing top 20 rows
```

3. user_profile

- 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

In [13]:
```
_user_profile_df = spark.read.csv('/data/user_profile.csv',header=True)
_user_profile_df.printSchema()
_user_profile_df.show()
```

```
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)

+------+---------+-----------+-----------------+---------+-----------+--------------+
----------+---------------------+
|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 [14]:

```
# 查看每列数据中有没有 'NULL',如果有,就不能使用schema,否则就会使那一整行就变成 null;如
# 注意:"null" 与 "NULL"
[str(c) + ':' + str(_user_profile_df.groupBy(c).count().show()) for c in _user_profil
# 根据结果可见仅仅有null,而没有NULL,因此可以使用schema
```

```
+-------+-----+
| userid|count|
+-------+-----+
| 505039|    1|
| 577511|    1|
| 627835|    1|
| 692974|    1|
| 742322|    1|
| 746750|    1|
| 777511|    1|
| 800757|    1|
| 878358|    1|
| 976473|    1|
|1141237|    1|
|  34635|    1|
| 265095|    1|
| 308633|    1|
| 344922|    1|
| 472235|    1|
| 618216|    1|
```

In [15]:
```
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_df1 = spark.read.csv('/data/user_profile.csv',header=True,schema=schema
_user_profile_df1.printSchema()
_user_profile_df1.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_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 [16]:
```
# 缺失值进行独热编码 pvalue_level列和new_user_class_level列
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_df2 = _user_profile_df1.fillna(-1)
_user_profile_df3 = _user_profile_df2.withColumn('pvalue_level',_user_profile_df2.pva
    withColumn('new_user_class_level',_user_profile_df2.new_user_class_level.cast(Stri

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_df3)
_user_profile_df4 = pipeline_fit.transform(_user_profile_df3)

stringindexer = StringIndexer(inputCol='new_user_class_level', outputCol='nucl_onehot
encoder = OneHotEncoder(dropLast=False, inputCol='nucl_onehot_feature', outputCol='n
pipeline = Pipeline(stages=[stringindexer, encoder])
pipeline_fit = pipeline.fit(_user_profile_df4)
user_profile_df = pipeline_fit.transform(_user_profile_df4)
user_profile_df.show()
```

```
+------+---------+-----------+----------------+---------+-----------+--------------+
----------+------------------+----------------+--------------+------------------+-
---------------+
|userId|cms_segid|cms_group_id|final_gender_code|age_level|pvalue_level|shopping_level|
occupation|new_user_class_level|pl_onehot_feature|pl_onehot_value|nucl_onehot_feature|n
ucl_onehot_value|
+------+---------+-----------+----------------+---------+-----------+--------------+
----------+------------------+----------------+--------------+------------------+-
---------------+
|   234|        0|          5|               2|        5|         -1|             3|
0|                 3|             0.0|  (4,[0],[1.0])|             2.0|    (5,
[2],[1.0])|
|   523|        5|          2|               2|        2|          1|             3|
1|                 2|             2.0|  (4,[2],[1.0])|             1.0|    (5,
[1],[1.0])|
|   612|        0|          8|               1|        2|          2|             3|
0|                -1|             1.0|  (4,[1],[1.0])|             0.0|    (5,
[0],[1.0])|
|  1670|        0|          4|               2|        4|         -1|             1|
0|                -1|             0.0|  (4,[0],[1.0])|             0.0|    (5,
[0],[1.0])|
|  2545|        0|         10|               1|        4|         -1|             3|
0|                -1|             0.0|  (4,[0],[1.0])|             0.0|    (5,
[0],[1.0])|
|  3644|       49|          6|               2|        6|          2|             3|
0|                 2|             1.0|  (4,[1],[1.0])|             1.0|    (5,
[1],[1.0])|
|  5777|       44|          5|               2|        5|          2|             3|
0|                 2|             1.0|  (4,[1],[1.0])|             1.0|    (5,
[1],[1.0])|
|  6211|        0|          9|               1|        3|         -1|             3|
0|                 2|             0.0|  (4,[0],[1.0])|             1.0|    (5,
[1],[1.0])|
|  6355|        2|          1|               2|        1|          1|             3|
0|                 4|             2.0|  (4,[2],[1.0])|             3.0|    (5,
```

```
[3],[1.0])|
|  6823|       43|          5|                2|        5|          2|               3|
0|                  1|             1.0|  (4,[1],[1.0])|                4.0|    (5,
[4],[1.0])|
|  6972|        5|          2|                2|        2|          2|               3|
1|                  2|             1.0|  (4,[1],[1.0])|                1.0|    (5,
[1],[1.0])|
|  9293|        0|          5|                2|        5|         -1|               3|
0|                  4|             0.0|  (4,[0],[1.0])|                3.0|    (5,
[3],[1.0])|
|  9510|       55|          8|                1|        2|          2|               2|
0|                  2|             1.0|  (4,[1],[1.0])|                1.0|    (5,
[1],[1.0])|
| 10122|       33|          4|                2|        4|          2|               3|
0|                  2|             1.0|  (4,[1],[1.0])|                1.0|    (5,
[1],[1.0])|
| 10549|        0|          4|                2|        4|          2|               3|
0|                 -1|             1.0|  (4,[1],[1.0])|                0.0|    (5,
[0],[1.0])|
| 10812|        0|          4|                2|        4|         -1|               2|
0|                 -1|             0.0|  (4,[0],[1.0])|                0.0|    (5,
[0],[1.0])|
| 10912|        0|          4|                2|        4|          2|               3|
0|                 -1|             1.0|  (4,[1],[1.0])|                0.0|    (5,
[0],[1.0])|
| 10996|        0|          5|                2|        5|         -1|               3|
0|                  4|             0.0|  (4,[0],[1.0])|                3.0|    (5,
[3],[1.0])|
| 11256|        8|          2|                2|        2|          1|               3|
0|                  3|             2.0|  (4,[2],[1.0])|                2.0|    (5,
[2],[1.0])|
| 11310|       31|          4|                2|        4|          1|               3|
0|                  4|             2.0|  (4,[2],[1.0])|                3.0|    (5,
[3],[1.0])|
+------+---------+-----------+-----------------+---------+-----------+--------------+
---------+-------------------+-----------------+--------------+------------------+-
---------------+
only showing top 20 rows
```

In [17]:
```
# 找出两者的映射关系 max min 都是一个值!
user_profile_df.groupby('pvalue_level').max('pl_onehot_feature').show()
user_profile_df.groupBy("new_user_class_level").max("nucl_onehot_feature").show()
```

```
+------------+----------------------+
|pvalue_level|max(pl_onehot_feature)|
+------------+----------------------+
|          -1|                   0.0|
|           3|                   3.0|
|           1|                   2.0|
|           2|                   1.0|
+------------+----------------------+

+-------------------+-----------------------+
|new_user_class_level|max(nucl_onehot_feature)|
+-------------------+-----------------------+
|                 -1|                    0.0|
|                  3|                    2.0|
|                  1|                    4.0|
|                  4|                    3.0|
|                  2|                    1.0|
+-------------------+-----------------------+
```

4.raw_sample表(包含userId和广告Id) 合并user_profile和ad_feature表

Dataframe数据合并： [pyspark.sql.DataFrame.join (https://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=join#pyspark.sql.DataFrame.join)](https://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=join#pyspark.sql.DataFrame.join)

[不同合并方式介绍 (https://stackoverflow.com/questions/38549/what-is-the-difference-between-inner-join-and-outer-join)](https://stackoverflow.com/questions/38549/what-is-the-difference-between-inner-join-and-outer-join)

In [28]:
```
# 由此可见 三张表的useid和adid个数是不一样的
print(raw_sample_df.count())
print(ad_feature_df.count())
print(user_profile_df.count())
print('*'*10)
print(raw_sample_df.groupBy('adgroupId').count().count())
print(ad_feature_df.groupBy('adgroupId').count().count())
print(raw_sample_df.groupBy('userId').count().count())
print(user_profile_df.groupBy('userId').count().count())
```

```
26557961
846811
1061768
**********
846811
846811
1141729
1061768
```

In [20]:
```
condition = [raw_sample_df.adgroupId == ad_feature_df.adgroupId]
_ = raw_sample_df.join(ad_feature_df,on=condition,how='outer')

condition2 = [_.userId == user_profile_df.userId]
datasets = _.join(user_profile_df,on=condition2,how='outer')

datasets.printSchema()
print(datasets.count())
```

```
root
 |-- userId: integer (nullable = true)
 |-- timestamp: long (nullable = true)
 |-- adgroupId: integer (nullable = true)
 |-- pid: string (nullable = true)
 |-- nonclk: integer (nullable = true)
 |-- clk: integer (nullable = true)
 |-- pid_feature: double (nullable = true)
 |-- pid_value: vector (nullable = true)
 |-- adgroupId: integer (nullable = true)
 |-- cateId: integer (nullable = true)
 |-- campaignId: integer (nullable = true)
 |-- customerId: integer (nullable = true)
 |-- brandId: integer (nullable = true)
 |-- price: float (nullable = true)
 |-- 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)

26557961
```

1. 训练CTRModel_Normal：直接将对应的特征的特征值组合成对应的特征向量进行训练

In [21]:
```
# 延申学习：作为条件的那些列不能被select
# datasets.select('nucl_onehot_feature')
```

In [23]:
```
# 剔除冗余、不需要的字段
useful_cols = [
    # 时间字段，划分训练集和测试集
    "timestamp",
    # label目标值字段
    "clk",
    # 特征值字段
    "pid_value",        # 资源位的特征向量
    "price",    # 广告价格
    "cms_segid",    # 用户微群ID
    "cms_group_id",    # 用户组ID
    "final_gender_code",    # 用户性别特征，[1,2]
    "age_level",    # 年龄等级，1-
    "shopping_level",
    "occupation",
    "pl_onehot_value",
    "nucl_onehot_value"
]
datasets_1 = datasets.select(*[useful_cols])
datasets_1.printSchema()
```

```
root
 |-- timestamp: long (nullable = true)
 |-- clk: integer (nullable = true)
 |-- pid_value: vector (nullable = true)
 |-- price: float (nullable = true)
 |-- cms_segid: integer (nullable = true)
 |-- cms_group_id: integer (nullable = true)
 |-- final_gender_code: integer (nullable = true)
 |-- age_level: integer (nullable = true)
 |-- shopping_level: integer (nullable = true)
 |-- occupation: integer (nullable = true)
 |-- pl_onehot_value: vector (nullable = true)
 |-- nucl_onehot_value: vector (nullable = true)
```

In [25]:
```
# 三张表行数不同,合并后肯定有空值,要去掉空值
# str类型的空值(null或者NULL)不能被dropna()掉
# str类型的NULL转化为 非str类型后,show()会显示null
datasets_1=datasets_1.dropna()
print("剔除空值数据后，还剩：", datasets_1.count())
```

剔除空值数据后，还剩： 25029435

根据特征字段计算出特征向量，并划分出训练数据集和测试数据集

In [ ]:
```
#### 根据特征字段计算出特征向量，并划分出训练数据集和测试数据集# 延申学习:找到最大时间
# datasets_1.orderBy('timestamp',ascending=False).show()
# #1494691186
```

In [32]:
```
# 根据特征字段 计算出特征向量, 并划分出 训练数据集合测试数据集
from pyspark.ml.feature import VectorAssembler
datasets_1 = VectorAssembler().setInputCols(useful_cols[2:]).setOutputCol('features').
#训练数据集
train_datasets_1 = datasets_1.filter(datasets_1.timestamp<=(1494691186-24*60*60))
#测试数据集
test_datasets_1 = datasets_1.where(datasets_1.timestamp>(1494691186-24*60*60))
# 所有特征的特征向量已经汇总在features字段中
train_datasets_1.show(5)
test_datasets_1.show(5)
```

...

创建逻辑回归训练器，并训练模型：LogisticRegression (https://spark.apache.org/docs/latest/api/python/pyspark.ml.html?highlight=logisticregression#pyspark.ml.classification.LogisticRegression)、LogisticRegressionModel (https://spark.apache.org/docs/latest/api/python/pyspark.ml.html?highlight=logisticregression#pyspark.ml.classification.LogisticRegressionModel)

In [33]:
```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression()
model = lr.setLabelCol('clk').setFeaturesCol('features').fit(train_datasets_1)
# model.save(hadoop上)

# 练习使用下面的类,使用已经训练好的模型
from pyspark.ml.classification import LogisticRegressionModel
model = LogisticRegressionModel.load('/models/CTRModel_Normal.obj')
result_1 = model.transform(test_datasets_1)
result_1.show()
```

...

In [34]:
```
result_1.select('clk','price','probability','prediction').sort('probability').show(100
# 预测的前20个,命中了3个
```

...

In [35]:
```
# 只查看样本中点击的被实际点击的条目的预测情况
result_1.select('clk','price','probability','prediction').where('clk==1').sort('probab
# 默认按照概率的50%进行分类,大于0.5预测为1,小于0.5预测为0
```

...

2. 训练CTRModel_AllOneHot

- "pid_value", 类别型特征，已被转换为多维特征==> 2维
- "price", 统计型特征 ===> 1维
- "cms_segid", 类别型特征，约97个分类 ===> 1维
- "cms_group_id", 类别型特征，约13个分类 ==> 1维
- "final_gender_code", 类别型特征，2个分类 ==> 1维
- "age_level", 类别型特征，7个分类 ==> 1维
- "shopping_level", 类别型特征，3个分类 ==> 1维
- "occupation", 类别型特征，2个分类 ==> 1维
- "pl_onehot_value", 类别型特征，已被转换为多维特征 ==> 4维

- "nucl_onehot_value" 类别型特征，已被转换为多维特征 ==> 5维

类别性特征都可以考虑进行热独编码，将单一变量变为多变量，相当于增加了相关特征的数量

- "cms_segid", 类别型特征，约97个分类 ===> 97维 舍弃
- "cms_group_id", 类别型特征，约13个分类 ==> 13维
- "final_gender_code", 类别型特征，2个分类 ==> 2维
- "age_level", 类别型特征，7个分类 ==>7维
- "shopping_level", 类别型特征，3个分类 ==> 3维
- "occupation", 类别型特征，2个分类 ==> 2维

但由于cms_segid分类过多，这里考虑舍弃，避免数据过于稀疏

In [42]:
```
# 首先查看每个特征列 有几个不同的值，n个不同的值就可以转换为n维
#=======================耗时4min==============================
[str(c) + '特征的种类个数:' + str(datasets_1.groupby(c).count().count()) for c in dat
```

Out[42]:
```
['pid_value特征的种类个数:2',
 'cms_segid特征的种类个数:97',
 'cms_group_id特征的种类个数:13',
 'final_gender_code特征的种类个数:2',
 'age_level特征的种类个数:7',
 'shopping_level特征的种类个数:3',
 'occupation特征的种类个数:2',
 'pl_onehot_value特征的种类个数:4',
 'nucl_onehot_value特征的种类个数:5']
```

In [43]:
```
datasets_1.first()
```

Out[43]:
```
Row(timestamp=1494261938, clk=0, pid_value=SparseVector(2, {1: 1.0}), price=108.0, cms_
segid=0, cms_group_id=11, final_gender_code=1, age_level=5, shopping_level=3, occupatio
n=0, pl_onehot_value=SparseVector(4, {0: 1.0}), nucl_onehot_value=SparseVector(5, {1:
1.0}), features=SparseVector(18, {1: 1.0, 2: 108.0, 4: 11.0, 5: 1.0, 6: 5.0, 7: 3.0, 9:
1.0, 14: 1.0}))
```

In [53]:
```
# 先将下列五列数据转为字符串类型，以便于进行热独编码
# - "cms_group_id",    类别型特征，约13个分类 ==> 13
# - "final_gender_code", 类别型特征，2个分类 ==> 2
# - "age_level",     类别型特征，7个分类 ==>7
# - "shopping_level",     类别型特征，3个分类 ==> 3
# - "occupation",     类别型特征，2个分类 ==> 2
# # datasets 三张表合并生成的表
datasets_2 = datasets.withColumn("cms_group_id", datasets.cms_group_id.cast(StringType
    .withColumn("final_gender_code", datasets.final_gender_code.cast(StringType()))\
    .withColumn("age_level", datasets.age_level.cast(StringType()))\
    .withColumn("shopping_level", datasets.shopping_level.cast(StringType()))\
    .withColumn("occupation", datasets.occupation.cast(StringType()))
useful_cols_2 = [
    # 时间值，划分训练集和测试集
    "timestamp",
    # label目标值
    "clk",
    # 特征值
    "price",
    "cms_group_id",
    "final_gender_code",
    "age_level",
    "shopping_level",
    "occupation",
    "pid_value",
    "pl_onehot_value",
    "nucl_onehot_value"
]
datasets_2 = datasets_2.select(*useful_cols_2)
datasets_2 = datasets_2.dropna()
```

In [54]:
```
from pyspark.ml.feature import OneHotEncoder
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline

def oneHotEncoder(col1,col2,col3,data):
    stringindexer = StringIndexer(inputCol=col1,outputCol=col2)
    encoder = OneHotEncoder(dropLast=False,inputCol=col2,outputCol=col3)
    pipeline = Pipeline(stages=[stringindexer,encoder])
    pipeline_fit = pipeline.fit(data)
    return pipeline_fit.transform(data)
# 对以下5个特征进行独热编码
# - "cms_group_id",    类别型特征，约13个分类 ==> 13
# - "final_gender_code", 类别型特征，2个分类 ==> 2
# - "age_level",     类别型特征，7个分类 ==>7
# - "shopping_level",     类别型特征，3个分类 ==> 3
# - "occupation",     类别型特征，2个分类 ==> 2
datasets_2 = oneHotEncoder("cms_group_id", "cms_group_id_feature", "cms_group_id_valu
datasets_2 = oneHotEncoder("final_gender_code", "final_gender_code_feature", "final_g
datasets_2 = oneHotEncoder("age_level", "age_level_feature", "age_level_value", datas
datasets_2 = oneHotEncoder("shopping_level", "shopping_level_feature", "shopping_leve
datasets_2 = oneHotEncoder("occupation", "occupation_feature", "occupation_value", da
```

In [55]:
```
# onehot编码完成后,查看下对应关系
# min max都是一个值,比如 1的min和max是一个值
datasets_2.groupBy("cms_group_id").min("cms_group_id_feature").show()
datasets_2.groupBy("final_gender_code").min("final_gender_code_feature").show()
datasets_2.groupBy("age_level").min("age_level_feature").show()
datasets_2.groupBy("shopping_level").min("shopping_level_feature").show()
datasets_2.groupBy("occupation").min("occupation_feature").show()
```

...

In [ ]:
```
# 独热编码后,特征字段不再是之前的字段,重新定义字段
feature_cols = [
    # 特征值
    "price",
    "cms_group_id_value",
    "final_gender_code_value",
    "age_level_value",
    "shopping_level_value",
    "occupation_value",
    "pid_value",
    "pl_onehot_value",
    "nucl_onehot_value"
]
# 根据特征字段计算出特征向量，并划分出训练数据集和测试数据集
from pyspark.ml.feature import VectorAssembler
datasets_2 = VectorAssembler().setInputCols(feature_cols).setOutputCol('features').tra
```

In [64]:
```
# 训练样本集
train_datasets_2 = datasets_2.filter(datasets_2.timestamp<=(1494691186-24*60*60))
# 测试样本集
test_datasets_2 = datasets_2.where(datasets_2.timestamp>(1494691186-24*60*60))
train_datasets_2.printSchema()
train_datasets_2.first()
# features=SparseVector(39, {0: 108.0, 7: 1.0, 15: 1.0, 18: 1.0, 23: 1.0, 26: 1.0, 29:
```

```
root
 |-- timestamp: long (nullable = true)
 |-- clk: integer (nullable = true)
 |-- price: float (nullable = true)
 |-- cms_group_id: string (nullable = true)
 |-- final_gender_code: string (nullable = true)
 |-- age_level: string (nullable = true)
 |-- shopping_level: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- pid_value: vector (nullable = true)
 |-- pl_onehot_value: vector (nullable = true)
 |-- nucl_onehot_value: vector (nullable = true)
 |-- cms_group_id_feature: double (nullable = true)
 |-- cms_group_id_value: vector (nullable = true)
 |-- final_gender_code_feature: double (nullable = true)
 |-- final_gender_code_value: vector (nullable = true)
 |-- age_level_feature: double (nullable = true)
 |-- age_level_value: vector (nullable = true)
 |-- shopping_level_feature: double (nullable = true)
 |-- shopping_level_value: vector (nullable = true)
 |-- occupation_feature: double (nullable = true)
 |-- occupation_value: vector (nullable = true)
 |-- features: vector (nullable = true)
```

Out[64]:
```
Row(timestamp=1494261938, clk=0, price=108.0, cms_group_id='11', final_gender_code='1',
age_level='5', shopping_level='3', occupation='0', pid_value=SparseVector(2, {1: 1.0}),
pl_onehot_value=SparseVector(4, {0: 1.0}), nucl_onehot_value=SparseVector(5, {1: 1.0}),
cms_group_id_feature=6.0, cms_group_id_value=SparseVector(13, {6: 1.0}), final_gender_c
ode_feature=1.0, final_gender_code_value=SparseVector(2, {1: 1.0}), age_level_feature=
2.0, age_level_value=SparseVector(7, {2: 1.0}), shopping_level_feature=0.0, shopping_le
vel_value=SparseVector(3, {0: 1.0}), occupation_feature=0.0, occupation_value=SparseVec
tor(2, {0: 1.0}), features=SparseVector(39, {0: 108.0, 7: 1.0, 15: 1.0, 18: 1.0, 23: 1.
0, 26: 1.0, 29: 1.0, 30: 1.0, 35: 1.0}))
```

In [67]:
```
#=================时间:=====================
# 创建逻辑回归训练器,并训练模型
from pyspark.ml.classification import LogisticRegression
lr2 = LogisticRegression()
model2 = lr2.setLabelCol('clk').setFeaturesCol('features').fit(train_datasets_2)
# models.save(hadoop上)
from pyspark.ml.classification import LogisticRegressionModel
model2 = LogisticRegressionModel.load('/models/CTRModel_AllOneHot.obj')
result_2 = model2.transform(test_datasets_2)
result_2.select('clk','price','probability','prediction').sort('probability').show(100
# 对比前面的result_1的预测结果，能发现这里的预测率稍微准确了一点，这里top20里出现了3个
# 因此可见对特征的细化处理，已经帮助我们提高模型的效果的
```

```
+---+----------+----------------------------------------+----------+
|clk|price     |probability                             |prediction|
+---+----------+----------------------------------------+----------+
|0  |1.0E8     |[0.855244188928558,0.1447558110714421]  |0.0       |
|0  |1.0E8     |[0.883531437621234,0.11646856237876606] |0.0       |
|0  |1.0E8     |[0.8916980898561577,0.10830191014384229]|0.0       |
|1  |5.5555556E7|[0.9251174396034961,0.07488256039650386]|0.0       |
|0  |179.01    |[0.9323995173830968,0.0676004826169032] |0.0       |
|1  |159.0     |[0.9323995290566156,0.06760047094338446]|0.0       |
|0  |118.0     |[0.9323995529753702,0.06760044702462979]|0.0       |
|0  |688.0     |[0.9345150616595344,0.0654849383404656] |0.0       |
|0  |339.0     |[0.9345152593362689,0.0654847406637311] |0.0       |
|0  |335.0     |[0.9345152616019017,0.06548473839809842]|0.0       |
|0  |220.0     |[0.9345153267388108,0.06548467326118919]|0.0       |
|0  |176.0     |[0.9345153516607427,0.06548464833925725]|0.0       |
|0  |158.0     |[0.9345153618560761,0.065484638143924]  |0.0       |
|0  |158.0     |[0.9345153618560761,0.065484638143924]  |0.0       |
|1  |149.0     |[0.934515366953742,0.06548463304625793] |0.0       |
|0  |122.5     |[0.9345153819635345,0.06548461803646553]|0.0       |
```

In [68]:
```
result_2.where('clk=1').select("clk", "price", "probability", "prediction").orderBy("p
# 从该结果也可以看出，result_2的点击率预测率普遍要比result_1高出一点点
```

```
+---+----------+----------------------------------------+----------+
|clk|price     |probability                             |prediction|
+---+----------+----------------------------------------+----------+
|1  |5.5555556E7|[0.9251174396034961,0.07488256039650386]|0.0       |
|1  |159.0     |[0.9323995290566156,0.06760047094338446]|0.0       |
|1  |149.0     |[0.934515366953742,0.06548463304625793] |0.0       |
|1  |8888.0    |[0.9349439274648473,0.0650560725351527] |0.0       |
|1  |138.0     |[0.9349441477080421,0.065055852291958]  |0.0       |
|1  |35.0      |[0.9349442056925659,0.06505579430743408]|0.0       |
|1  |519.0     |[0.934948638706219,0.06505136129378104] |0.0       |
|1  |478.0     |[0.9349486617859604,0.06505133821403952]|0.0       |
|1  |349.0     |[0.9349487344026585,0.06505126559734156]|0.0       |
|1  |348.0     |[0.9349487349655783,0.06505126503442173]|0.0       |
|1  |316.0     |[0.9349487529790108,0.06505124702098909]|0.0       |
|1  |298.0     |[0.9349487631115648,0.06505123688843525]|0.0       |
|1  |298.0     |[0.9349487631115648,0.06505123688843525]|0.0       |
|1  |199.0     |[0.9349488188405846,0.06505118115941552]|0.0       |
|1  |199.0     |[0.9349488188405846,0.06505118115941552]|0.0       |
|1  |198.0     |[0.9349488194035036,0.06505118059649637]|0.0       |
```

In [ ]:
```
1
```