京东JData大数据竞赛（1）- 数据清洗

比赛的题目是高潜用户的购买意向的预测,从机器学习的角度来讲我们可以认为这是一个二分类的任务.那么我们就是尝试去构建自己的正负样本. 由于我们拿到的是原始数据,里面存在很多噪声,因而第一步我们先要对数据清洗,比如说:

- 检查并去除重复记录
- 无交互行为的用户和商品
- 去掉浏览量很大而购买量很少的用户(惰性用户或爬虫用户)
- ……

为了能够进行上述清洗,在此首先构造了简单的用户(user)行为特征和商品(item)行为特征,对应于两张表user_table和item_table

- user_table特征包括:
  user_id(用户id),age(年龄),sex(性别),
  user_lv_cd(用户级别),browse_num(浏览数),
  addcart_num(加购数),delcart_num(删购数),
  buy_num(购买数),favor_num(收藏数),
  click_num(点击数),buy_addcart_ratio(购买加购转化率),
  buy_browse_ratio(购买浏览转化率),
  buy_click_ratio(购买点击转化率),
  buy_favor_ratio(购买收藏转化率)
- item_table特征包括:
  sku_id(商品id),attr1,attr2,
  attr3,cate,brand,browse_num,
  addcart_num,delcart_num,
  buy_num,favor_num,click_num,
  buy_addcart_ratio,buy_browse_ratio,
  buy_click_ratio,buy_favor_ratio,
  comment_num(评论数),
  has_bad_comment(是否有差评),
  bad_comment_rate(差评率)

数据集注释

这里涉及到的数据集是京东最新的数据集：

1. JData_User.csv 用户数据集 105,321个用户
2. JData_Comment.csv 商品评论 558,552条记录
3. JData_Product.csv 预测商品集合 24,187条记录
4. JData_Action_201602.csv 2月份行为交互记录 11,485,424条记录
5. JData_Action_201603.csv 3月份行为交互记录 25,916,378条记录
6. JData_Action_201604.csv 4月份行为交互记录 13,199,934条记录

数据集验证

首先检查JData_User中的用户和JData_Action中的用户是否一致

保证行为数据中的所产生的行为均由用户数据中的用户产生（但是可能存在用户在行为数据中无行为）

思路：利用pd.Merge连接sku 和 Action中的sku, 观察Action中的数据是否减少 Example:

```
import pandas as pd
# test sample
# df1 = pd.DataFrame({'sku':['a','a','b','c'],'data':[1,1,2,3]})
# df2 = pd.DataFrame({'sku':['a','b','c']})
# df3 = pd.DataFrame({'sku':['a','b','d']})
# df4 = pd.DataFrame({'sku':['a','b','c','d']})
# print pd.merge(df2,df1)
# # print pd.merge(df1,df2)
# print pd.merge(df3,df1)
# print pd.merge(df4,df1)
# # print pd.merge(df1,df3)
```

```
def user_action_check():
    df_user = pd.read_csv('data/JData_User.csv')
    df_sku = df_user.ix[:,'user_id'].to_frame()
    df_month2 = pd.read_csv('data/JData_Action_201602.csv')
    print 'Is action of Feb. from User file? ', len(df_month2) == len(pd.merge(df_sku,df_month2))
    df_month3 = pd.read_csv('data/JData_Action_201603.csv')
    print 'Is action of Mar. from User file? ', len(df_month3) == len(pd.merge(df_sku,df_month3))
    df_month4 = pd.read_csv('data/JData_Action_201604.csv')
    print 'Is action of Apr. from User file? ', len(df_month4) == len(pd.merge(df_sku,df_month4))

user_action_check()
```

```
Is action of Feb. from User file?  True
Is action of Mar. from User file?  True
Is action of Apr. from User file?  True
```

结论： User数据集中的用户和交互行为数据集中的用户完全一致

根据merge前后的数据量比对，能保证Action中的用户ID是User中的ID的子集

检查是否有重复记录

除去各个数据文件中完全重复的记录,结果证明线上成绩反而大幅下降，可能解释是重复数据是有意义的，比如用户同时购买多件商品，同时添加多个数量的商品到购物车等...

```
def deduplicate(filepath, filename, newpath):
    df_file = pd.read_csv(filepath)
    before = df_file.shape[0]
    df_file.drop_duplicates(inplace=True)
    after = df_file.shape[0]
    n_dup = before-after
    print 'No. of duplicate records for ' + filename + ' is: ' + str(n_dup)
    if n_dup != 0:
        df_file.to_csv(newpath, index=None)
    else:
        print 'no duplicate records in ' + filename
```

```
# deduplicate('data/JData_Action_201602.csv', 'Feb. action', 'data/JData_Action_201602_dedup.csv')
deduplicate('data/JData_Action_201603.csv', 'Mar. action', 'data/JData_Action_201603_dedup.csv')
deduplicate('data/JData_Action_201604.csv', 'Feb. action', 'data/JData_Action_201604_dedup.csv')
deduplicate('data/JData_Comment.csv', 'Comment', 'data/JData_Comment_dedup.csv')
deduplicate('data/JData_Product.csv', 'Product', 'data/JData_Product_dedup.csv')
deduplicate('data/JData_User.csv', 'User', 'data/JData_User_dedup.csv')
```

```
No. of duplicate records for Mar. action is: 7085038
No. of duplicate records for Feb. action is: 3672710
No. of duplicate records for Comment is: 0
no duplicate records in Comment
No. of duplicate records for Product is: 0
no duplicate records in Product
No. of duplicate records for User is: 0
no duplicate records in User
```

```
IsDuplicated = df_month.duplicated()
df_d=df_month[IsDuplicated]
df_d.groupby('type').count()              #发现重复数据大多数都是由于浏览（1），或者点击(6)产生
```

	user_id	sku_id	time	model_id	cate	brand
type						
1	2176378	2176378	2176378	0	2176378	2176378
2	636	636	636	0	636	636
3	1464	1464	1464	0	1464	1464
4	37	37	37	0	37	37
5	1981	1981	1981	0	1981	1981
6	575597	575597	575597	545054	575597	575597

检查是否存在注册时间在2016年-4月-15号之后的用户

```
import pandas as pd
df_user = pd.read_csv('data\JData_User.csv',encoding='gbk')
df_user['user_reg_tm']=pd.to_datetime(df_user['user_reg_tm'])
df_user.ix[df_user.user_reg_tm  >= '2016-4-15']
```

	user_id	age	sex	user_lv_cd	user_reg_tm
7457	207458	-1	2.0	1	2016-04-15
7463	207464	26-35岁	2.0	2	2016-04-15
7467	207468	36-45岁	2.0	3	2016-04-15
7472	207473	-1	2.0	1	2016-04-15
7482	207483	26-35岁	2.0	3	2016-04-15
7492	207493	16-25岁	2.0	3	2016-04-15
7493	207494	16-25岁	2.0	3	2016-04-15
7503	207504	16-25岁	2.0	4	2016-04-15
7510	207511	46-55岁	2.0	5	2016-04-15
7512	207513	-1	2.0	1	2016-04-15
7518	207519	26-35岁	2.0	2	2016-04-15
7521	207522	26-35岁	0.0	3	2016-04-15
7525	207526	-1	2.0	3	2016-04-15
7533	207534	-1	2.0	1	2016-04-15
7543	207544	26-35岁	2.0	3	2016-04-15
7544	207545	-1	2.0	1	2016-04-15
7551	207552	26-35岁	2.0	3	2016-04-15
7553	207554	16-25岁	2.0	4	2016-04-15
8545	208546	16-25岁	0.0	2	2016-04-29
9394	209395	16-25岁	1.0	2	2016-05-11
10362	210363	56岁以上	2.0	2	2016-05-24
10367	210368	-1	2.0	1	2016-05-24
11019	211020	36-45岁	2.0	3	2016-06-06
12014	212015	36-45岁	2.0	2	2016-07-05
13850	213851	26-35岁	2.0	3	2016-09-11
14542	214543	-1	2.0	1	2016-10-05
16746	216747	16-25岁	2.0	1	2016-11-25

由于注册时间是京东系统错误造成，如果行为数据中没有在4月15号之后的数据的话，那么说明这些用户还是正常用户，并不需要删除。

```
df_month = pd.read_csv('data\JData_Action_201604.csv')
df_month['time'] = pd.to_datetime(df_month['time'])
df_month.ix[df_month.time >= '2016-4-16']
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  This is separate from the ipykernel package so we can avoid doing imports until
```

	user_id	sku_id	time	model_id	type	cate	brand

结论：说明用户没有异常操作数据，所以这一批用户不删除

行为数据中的user_id为浮点型，进行INT类型转换

```
import pandas as pd
df_month = pd.read_csv('data\JData_Action_201602.csv')
df_month['user_id'] = df_month['user_id'].apply(lambda x:int(x))
print(df_month['user_id'].dtype)
df_month.to_csv('data\JData_Action_201602.csv',index=None)
df_month = pd.read_csv('data\JData_Action_201603.csv')
df_month['user_id'] = df_month['user_id'].apply(lambda x:int(x))
print(df_month['user_id'].dtype)
df_month.to_csv('data\JData_Action_201603.csv',index=None)
df_month = pd.read_csv('data\JData_Action_201604.csv')
df_month['user_id'] = df_month['user_id'].apply(lambda x:int(x))
print(df_month['user_id'].dtype)
df_month.to_csv('data\JData_Action_201604.csv',index=None)
```

```
int64
int64
int64
```

年龄区间的处理

```
import pandas as pd
df_user = pd.read_csv('data\JData_User.csv',encoding='gbk')

def tranAge(x):
    if x == u'15岁以下':
        x='1'
    elif x==u'16-25岁':
        x='2'
    elif x==u'26-35岁':
        x='3'
    elif x==u'36-45岁':
        x='4'
    elif x==u'46-55岁':
        x='5'
    elif x==u'56岁以上':
        x='6'
    return x
df_user['age']=df_user['age'].apply(tranAge)
print(df_user.groupby(df_user['age']).count())
df_user.to_csv('data\JData_User.csv',index=None)
```

```
     user_id    sex  user_lv_cd  user_reg_tm
age
-1     14412  14412       14412        14412
1          7      7           7            7
2       8797   8797        8797         8797
3      46570  46570       46570        46570
4      30336  30336       30336        30336
5       3325   3325        3325         3325
6       1871   1871        1871         1871
```

构建User_table

```
#定义文件名
ACTION_201602_FILE = "data/JData_Action_201602.csv"
ACTION_201603_FILE = "data/JData_Action_201603.csv"
ACTION_201604_FILE = "data/JData_Action_201604.csv"
COMMENT_FILE = "data/JData_Comment.csv"
PRODUCT_FILE = "data/JData_Product.csv"
USER_FILE = "data/JData_User.csv"
USER_TABLE_FILE = "data/User_table.csv"
ITEM_TABLE_FILE = "data/Item_table.csv"
```

```
# 导入相关包
import pandas as pd
import numpy as np
from collections import Counter
```

```
# 功能函数：对每一个user分组的数据进行统计
def add_type_count(group):
    behavior_type = group.type.astype(int)
    # 用户行为类别
    type_cnt = Counter(behavior_type)
    # 1：浏览 2：加购 3：删除
    # 4：购买 5：收藏 6：点击
    group['browse_num'] = type_cnt[1]
    group['addcart_num'] = type_cnt[2]
    group['delcart_num'] = type_cnt[3]
    group['buy_num'] = type_cnt[4]
    group['favor_num'] = type_cnt[5]
    group['click_num'] = type_cnt[6]

    return group[['user_id', 'browse_num', 'addcart_num',
                  'delcart_num', 'buy_num', 'favor_num',
                  'click_num']]
```

由于用户行为数据量较大,一次性读入可能造成内存错误(Memory Error),因而使用pandas的分块(chunk)读取.

```
#对action数据进行统计
#根据自己调节chunk_size大小
def get_from_action_data(fname, chunk_size=100000):
    reader = pd.read_csv(fname, header=0, iterator=True)
    chunks = []
    loop = True
    while loop:
        try:
            # 只读取user_id和type两个字段
            chunk = reader.get_chunk(chunk_size)[["user_id", "type"]]
            chunks.append(chunk)
        except StopIteration:
            loop = False
            print("Iteration is stopped")
    # 将块拼接为pandas dataframe格式
    df_ac = pd.concat(chunks, ignore_index=True)
    # 按user_id分组，对每一组进行统计，as_index 表示无索引形式返回数据
    df_ac = df_ac.groupby(['user_id'], as_index=False).apply(add_type_count)
    # 将重复的行丢弃
    df_ac = df_ac.drop_duplicates('user_id')

    return df_ac
```

```
# 将各个action数据的统计量进行聚合
def merge_action_data():
    df_ac = []
    df_ac.append(get_from_action_data(fname=ACTION_201602_FILE))
    df_ac.append(get_from_action_data(fname=ACTION_201603_FILE))
    df_ac.append(get_from_action_data(fname=ACTION_201604_FILE))

    df_ac = pd.concat(df_ac, ignore_index=True)
    # 用户在不同action表中统计量求和
    df_ac = df_ac.groupby(['user_id'], as_index=False).sum()
    #  构造转化率字段
    df_ac['buy_addcart_ratio'] = df_ac['buy_num'] / df_ac['addcart_num']
    df_ac['buy_browse_ratio'] = df_ac['buy_num'] / df_ac['browse_num']
    df_ac['buy_click_ratio'] = df_ac['buy_num'] / df_ac['click_num']
    df_ac['buy_favor_ratio'] = df_ac['buy_num'] / df_ac['favor_num']

    # 将大于1的转化率字段置为1(100%)
    df_ac.ix[df_ac['buy_addcart_ratio'] > 1., 'buy_addcart_ratio'] = 1.
    df_ac.ix[df_ac['buy_browse_ratio'] > 1., 'buy_browse_ratio'] = 1.
    df_ac.ix[df_ac['buy_click_ratio'] > 1., 'buy_click_ratio'] = 1.
    df_ac.ix[df_ac['buy_favor_ratio'] > 1., 'buy_favor_ratio'] = 1.

    return df_ac
```

```
#  从FJData_User表中抽取需要的字段
def get_from_jdata_user():
    df_usr = pd.read_csv(USER_FILE, header=0)
    df_usr = df_usr[["user_id", "age", "sex", "user_lv_cd"]]
    return df_usr
```

```
user_base = get_from_jdata_user()
user_behavior = merge_action_data()

# 连接成一张表，类似于SQL的左连接(left join)
user_behavior = pd.merge(user_base, user_behavior, on=['user_id'], how='left')
# 保存为user_table.csv
user_behavior.to_csv(USER_TABLE_FILE, index=False)
```

```
Iteration is stopped
Iteration is stopped
Iteration is stopped
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:18: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

构建Item_table

```
#定义文件名
ACTION_201602_FILE = "data/JData_Action_201602.csv"
ACTION_201603_FILE = "data/JData_Action_201603.csv"
ACTION_201604_FILE = "data/JData_Action_201604.csv"
COMMENT_FILE = "data/JData_Comment.csv"
PRODUCT_FILE = "data/JData_Product.csv"
USER_FILE = "data/JData_User.csv"
USER_TABLE_FILE = "data/User_table.csv"
ITEM_TABLE_FILE = "data/Item_table.csv"
```

```
# 导入相关包
import pandas as pd
import numpy as np
from collections import Counter
```

```
# 读取Product中商品
def get_from_jdata_product():
    df_item = pd.read_csv(PRODUCT_FILE, header=0)
    return df_item
```

```
# 对每一个商品分组进行统计
def add_type_count(group):
    behavior_type = group.type.astype(int)
    type_cnt = Counter(behavior_type)

    group['browse_num'] = type_cnt[1]
    group['addcart_num'] = type_cnt[2]
    group['delcart_num'] = type_cnt[3]
    group['buy_num'] = type_cnt[4]
    group['favor_num'] = type_cnt[5]
    group['click_num'] = type_cnt[6]

    return group[['sku_id', 'browse_num', 'addcart_num',
                  'delcart_num', 'buy_num', 'favor_num',
                  'click_num']]
```

```
#对action中的数据进行统计
def get_from_action_data(fname, chunk_size=100000):
    reader = pd.read_csv(fname, header=0, iterator=True)
    chunks = []
    loop = True
    while loop:
        try:
            chunk = reader.get_chunk(chunk_size)[["sku_id", "type"]]
            chunks.append(chunk)
        except StopIteration:
            loop = False

            print("Iteration is stopped")
```

```
    df_ac = pd.concat(chunks, ignore_index=True)

    df_ac = df_ac.groupby(['sku_id'], as_index=False).apply(add_type_count)
    # Select unique row
    df_ac = df_ac.drop_duplicates('sku_id')

    return df_ac
```

```
# 获取评论中的商品数据,如果存在某一个商品有两个日期的评论，我们取最晚的那一个
def get_from_jdata_comment():
    df_cmt = pd.read_csv(COMMENT_FILE, header=0)
    df_cmt['dt'] = pd.to_datetime(df_cmt['dt'])
    # find latest comment index
    idx = df_cmt.groupby(['sku_id'])['dt'].transform(max) == df_cmt['dt']
    df_cmt = df_cmt[idx]

    return df_cmt[['sku_id', 'comment_num',
                   'has_bad_comment', 'bad_comment_rate']]
```

```
def merge_action_data():
    df_ac = []
    df_ac.append(get_from_action_data(fname=ACTION_201602_FILE))
    df_ac.append(get_from_action_data(fname=ACTION_201603_FILE))
    df_ac.append(get_from_action_data(fname=ACTION_201604_FILE))

    df_ac = pd.concat(df_ac, ignore_index=True)
    df_ac = df_ac.groupby(['sku_id'], as_index=False).sum()

    df_ac['buy_addcart_ratio'] = df_ac['buy_num'] / df_ac['addcart_num']
    df_ac['buy_browse_ratio'] = df_ac['buy_num'] / df_ac['browse_num']
    df_ac['buy_click_ratio'] = df_ac['buy_num'] / df_ac['click_num']
    df_ac['buy_favor_ratio'] = df_ac['buy_num'] / df_ac['favor_num']

    df_ac.ix[df_ac['buy_addcart_ratio'] > 1., 'buy_addcart_ratio'] = 1.
    df_ac.ix[df_ac['buy_browse_ratio'] > 1., 'buy_browse_ratio'] = 1.
    df_ac.ix[df_ac['buy_click_ratio'] > 1., 'buy_click_ratio'] = 1.
    df_ac.ix[df_ac['buy_favor_ratio'] > 1., 'buy_favor_ratio'] = 1.

    return df_ac
```

```
#取出只有P集合中存在商品信息
item_base = get_from_jdata_product()
item_behavior = merge_action_data()
item_comment = get_from_jdata_comment()

# SQL: left join
item_behavior = pd.merge(
    item_base, item_behavior, on=['sku_id'], how='left')
item_behavior = pd.merge(
    item_behavior, item_comment, on=['sku_id'], how='left')

item_behavior.to_csv(ITEM_TABLE_FILE, index=False)
```

```
Iteration is stopped
Iteration is stopped
Iteration is stopped
```

数据清洗

用户清洗

```
import pandas as pd
df_user = pd.read_csv('data/User_table.csv',header=0)
pd.options.display.float_format = '{:,.3f}'.format  #输出格式设置，保留三位小数
df_user.describe()
```

	user_id	age	sex	user_lv_cd	browse_num	addcart_num	delcart_num	buy_num	favor_num
count	105,321.000	105,318.000	105,318.000	105,321.000	105,180.000	105,180.000	105,180.000	105,180.000	105,180.000
mean	252,661.000	2.773	1.113	3.850	180.466	5.471	2.434	0.459	1.045
std	30,403.698	1.672	0.956	1.072	273.437	10.618	5.600	1.048	3.442
min	200,001.000	-1.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
25%	226,331.000	3.000	0.000	3.000	40.000	0.000	0.000	0.000	0.000
50%	252,661.000	3.000	2.000	4.000	94.000	2.000	0.000	0.000	0.000
75%	278,991.000	4.000	2.000	5.000	212.000	6.000	3.000	1.000	0.000
max	305,321.000	6.000	2.000	5.000	7,605.000	369.000	231.000	50.000	99.000

由上述统计信息发现： 第一行中根据User_id统计发现有105321个用户，发现有3个用户没有age,sex字段，而且根据浏览、加购、删购、购买等记录却只有105180条记录，说明存在用户无任何交互记录，因此可以删除上述用户。

删除没有age,sex字段的用户

```
df_user[df_user['age'].isnull()]
```

	user_id	age	sex	user_lv_cd	browse_num	addcart_num	delcart_num	buy_num	favor_num	click_num	buy_
34072	234073	nan	nan	1	32.000	6.000	4.000	1.000	0.000	41.000	0.167
38905	238906	nan	nan	1	171.000	3.000	2.000	2.000	3.000	464.000	0.667
67704	267705	nan	nan	1	342.000	18.000	8.000	0.000	0.000	743.000	0.000

```
delete_list = df_user[df_user['age'].isnull()].index
df_user.drop(delete_list,axis=0,inplace=True)
```

删除无交互记录的用户

```
#删除无交互记录的用户
df_naction = df_user[(df_user['browse_num'].isnull()) & (df_user['addcart_num'].isnull()) &
(df_user['delcart_num'].isnull()) & (df_user['buy_num'].isnull()) & (df_user['favor_num'].isnull()) &
(df_user['click_num'].isnull())]
df_user.drop(df_naction.index,axis=0,inplace=True)
print len(df_user)
```

```
105177
```

统计并删除无购买记录的用户

```
#统计无购买记录的用户
df_bzero = df_user[df_user['buy_num']==0]
#输出购买数为0的总记录数
print len(df_bzero)
```

```
75694
```

```
#删除无购买记录的用户
df_user = df_user[df_user['buy_num']!=0]
```

```
df_user.describe()
```

	user_id	age	sex	user_lv_cd	browse_num	addcart_num	delcart_num	buy_num	favor_num
count	29,483.000	29,483.000	29,483.000	29,483.000	29,483.000	29,483.000	29,483.000	29,483.000	29,483.000
mean	250,746.445	2.914	1.025	4.272	302.488	10.525	4.673	1.637	1.677
std	29,979.676	1.490	0.959	0.808	391.535	14.301	7.568	1.412	4.584
min	200,001.000	-1.000	0.000	2.000	1.000	0.000	0.000	1.000	0.000
25%	225,058.500	3.000	0.000	4.000	76.000	3.000	0.000	1.000	0.000
50%	249,144.000	3.000	1.000	4.000	178.000	6.000	2.000	1.000	0.000
75%	276,252.500	4.000	2.000	5.000	381.000	13.000	6.000	2.000	1.000
max	305,318.000	6.000	2.000	5.000	7,605.000	288.000	178.000	50.000	96.000

删除爬虫及惰性用户

由上表所知，浏览购买转换比和点击购买转换比均值为0.018,0.030，因此这里认为浏览购买转换比和点击购买转换比小于0.0005的用户为惰性用户

```
bindex = df_user[df_user['buy_browse_ratio']<0.0005].index
print len(bindex)
df_user.drop(bindex,axis=0,inplace=True)
```

```
90
```

```
cindex = df_user[df_user['buy_click_ratio']<0.0005].index
print len(cindex)
df_user.drop(cindex,axis=0,inplace=True)
```

```
323
```

```
df_user.describe()
```

	user_id	age	sex	user_lv_cd	browse_num	addcart_num	delcart_num	buy_num	favor_num
count	29,070.000	29,070.000	29,070.000	29,070.000	29,070.000	29,070.000	29,070.000	29,070.000	29,070.000
mean	250,767.099	2.910	1.028	4.268	280.260	10.145	4.457	1.644	1.589
std	29,998.870	1.492	0.959	0.809	325.129	13.443	6.998	1.420	4.294
min	200,001.000	-1.000	0.000	2.000	1.000	0.000	0.000	1.000	0.000
25%	225,036.000	3.000	0.000	4.000	75.000	3.000	0.000	1.000	0.000
50%	249,200.500	3.000	1.000	4.000	174.000	6.000	2.000	1.000	0.000
75%	276,284.000	4.000	2.000	5.000	366.000	13.000	6.000	2.000	1.000
max	305,318.000	6.000	2.000	5.000	5,007.000	288.000	158.000	50.000	69.000

最后这29070个用户为最终预测用户数据集

```
df_user.to_csv("data/JData_FUser.csv",index=None)
```

商品清洗

```
import pandas as pd
df_product = pd.read_csv('data/Item_table.csv',header=0)
pd.options.display.float_format = '{:,.3f}'.format  #输出格式设置，保留三位小数
df_product.describe()
```

	sku_id	a1	a2	a3	cate	brand	browse_num	addcart_num	delcart_num	b
count	24,187.000	24,187.000	24,187.000	24,187.000	24,187.000	24,187.000	3,938.000	3,938.000	3,938.000	3
mean	85,398.737	2.177	0.939	1.180	8.000	435.864	1,723.711	54.212	21.284	3
std	49,238.799	1.176	0.970	1.046	0.000	225.749	7,957.661	285.723	106.100	2
min	6.000	-1.000	-1.000	-1.000	8.000	3.000	0.000	0.000	0.000	0
25%	42,476.000	1.000	1.000	1.000	8.000	214.000	19.000	0.000	0.000	0
50%	85,616.000	3.000	1.000	1.000	8.000	489.000	148.000	1.000	1.000	0
75%	127,774.000	3.000	2.000	2.000	8.000	571.000	655.750	12.000	5.000	0
max	171,224.000	3.000	2.000	2.000	8.000	922.000	194,920.000	6,296.000	2,258.000	6