淘宝用户行为分析

最终项目文档

Matrix
刘东洋 蔡依璇

目录：

一 项目概述

1.1 项目背景

大市场环境下，淘宝网是我国受欢迎的网购平台，拥有庞大的用户群体，每天的商品交易量巨大，是我国市场经济的主要部分。故而，如何提高淘宝店家的收益，不仅让企业收入提高，也让用户享受到更

高品质高质量的服务和消费体验是一项重大问题。

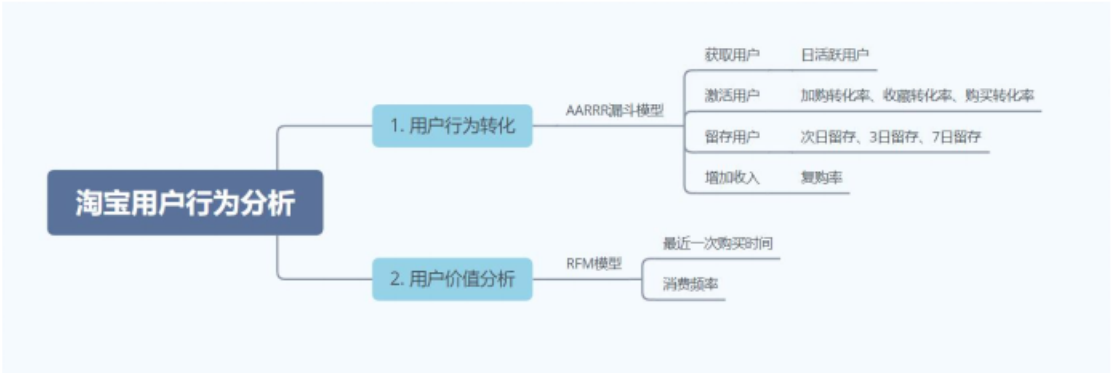
1.2 项目意义

通过对以往数据的分析，有助于企业根据用户的行为习惯，找出网站，平台和推广渠道等企业营销环境存在的问题，以此进行服务的改进，最终达到提高企业收入的目的；同时，通过企业分析数据对自己的服务进行整改，也为用户提高了服务和消费的质量和体验感。

二 项目思路框架

思路：针对数据集中的用户、商品、商品种类、用户行为、时间等信息，使用 Python 对数据进行切片分类汇总等多种数据分析手段，从不同角度挖掘蕴含的价值。

框架



三 具体分析过程

3.1 获得并整理数据

数据集：

本数据集随机采集了在 2017 年 11 月 25 日至 2017 年 12 月 3 日之间，淘宝用户的行为，其中行为包括浏览、加购物车、收藏、购买等。数据集主要包含：用户数量约 3 万（37,376），商品数量约 9 万（930,607），商品类目数量 7106 以及总的淘宝用户行为记录数量为 3 百万（3,835,329）。

数据来源：阿里云天池 https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1

数据集各字段含义：

列名称	说明
User ID	整数类型，序列化后的用户ID
Item ID	整数类型，序列化后的商品ID
Category ID	整数类型，序列化后的商品所属类目ID
Behavior type	字符串，枚举类型，包括('pv', 'buy', 'cart', 'fav')
Timestamp	行为发生的时间戳

行为类型	说明
pv	商品详情页pv，等价于点击
buy	商品购买
cart	将商品加入购物车
fav	收藏商品

3.2 数据清洗及特征工程

3.2.1 数据导入

首先导入数据分析和可视化所需的工具包

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
%matplotlib inline
```

然后读取数据集，将数据读入内存

```
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

用 data_user.head()查看数据集前五条数据

	UserID	ItemID	CatogoryID	BehaviorType	TimeStamps
0	1	2333346	2520771	pv	1511561733
1	1	2576651	149192	pv	1511572885
2	1	3830808	4181361	pv	1511593493
3	1	4365585	2520377	pv	1511596146
4	1	4606018	2735466	pv	1511616481

data_user.info()查看数据集字段信息和大小

```
data_user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100150806 entries, 0 to 100150805
Data columns (total 5 columns):
 #   Column        Dtype
---  ------        -----
 0   UserID        int64
 1   ItemID        int64
 2   CatogoryID    int64
 3   BehaviorType  object
 4   TimeStamps    int64
dtypes: int64(4), object(1)
memory usage: 3.7+ GB
```

data_user.describe()查看数据各字段的基本统计量

```
data_user.describe()
```

	UserID	ItemID	CatogoryID	TimeStamps
count	1.001508e+08	1.001508e+08	1.001508e+08	1.001508e+08
mean	5.069431e+05	2.579775e+06	2.696380e+06	1.511951e+09
std	2.940605e+05	1.488056e+06	1.463155e+06	5.528006e+06
min	1.000000e+00	1.000000e+00	8.000000e+01	-2.134949e+09
25%	2.524290e+05	1.295225e+06	1.320293e+06	1.511762e+09
50%	5.040150e+05	2.580735e+06	2.671397e+06	1.511965e+09
75%	7.609490e+05	3.862042e+06	4.145813e+06	1.512179e+09
max	1.018011e+06	5.163070e+06	5.162429e+06	2.122867e+09

3.2.2 缺失值分析

```
#缺失值分析
data_user.apply(lambda x: sum(x.isnull()))
```

```
UserID          0
ItemID          0
CatogoryID      0
BehaviorType    0
TimeStamps      0
dtype: int64
```

只有一条缺失值，做删除处理

3.2.3 时间格式化

选取符合需求的时间范围（2017-11-25 00:00:00 至 2017-12-3 23:59:59）

```
def get_unixtime(timeStr):
    formatStr = "%Y-%m-%d %H:%M:%S"
    tmObject = time.strptime(timeStr, formatStr)
    tmStamp = time.mktime(tmObject)

    return int(tmStamp)

# 数据集描述的时间范围
startTime = get_unixtime("2017-11-25 00:00:00")
endTime = get_unixtime("2017-12-3 23:59:59")

# 筛选出符合时间范围的数据
data_user['TimeStamps'] = data_user['TimeStamps'].astype('int64')
data_user = data_user.loc[(data_user['TimeStamps'] >= startTime) & (data_user['TimeStamps'] <= endTime)]
```

将时间分为 date,time,hour 三个字段

```
#时间处理（规范化）
data_user['time'] = data_user['TimeStamps'].apply(lambda t: time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(t)))
data_user['date'] = data_user['time'].str[0:10]
data_user['hour'] = data_user['time'].str[11:13].astype(int)
```

查看处理完成后的各字段和各字段信息

```
data_user.head()
```

	UserID	ItemID	CatogoryID	BehaviorType	TimeStamps	time	date	hour
0	1	2333346	2520771	pv	1511561733	2017-11-25 06:15:33	2017-11-25	6
1	1	2576651	149192	pv	1511572885	2017-11-25 09:21:25	2017-11-25	9
2	1	3830808	4181361	pv	1511593493	2017-11-25 15:04:53	2017-11-25	15
3	1	4365585	2520377	pv	1511596146	2017-11-25 15:49:06	2017-11-25	15
4	1	4606018	2735466	pv	1511616481	2017-11-25 21:28:01	2017-11-25	21

```
[21]: data_user.describe(include='all')
```

[21]:

	UserID	ItemID	CatogoryID	BehaviorType	TimeStamps	time	date	hour
count	3.833385e+06	3.833385e+06	3.833385e+06	3833385	3.833385e+06	3833385	3833385	3.833385e+06
unique	NaN	NaN	NaN	4	NaN	697064	9	NaN
top	NaN	NaN	NaN	pv	NaN	2017-11-26 22:53:07	2017-12-02 00:00:00	NaN
freq	NaN	NaN	NaN	3431904	NaN	55	532774	NaN
first	NaN	NaN	NaN	NaN	NaN	NaN	2017-11-25 00:00:00	NaN
last	NaN	NaN	NaN	NaN	NaN	NaN	2017-12-03 00:00:00	NaN
mean	2.411379e+05	2.578342e+06	2.711698e+06	NaN	1.511963e+09	NaN	NaN	1.490921e+01
std	2.548327e+05	1.487850e+06	1.464916e+06	NaN	2.301480e+05	NaN	NaN	6.095802e+00
min	1.000000e+00	3.000000e+00	2.171000e+03	NaN	1.511539e+09	NaN	NaN	0.000000e+00
25%	1.299830e+05	1.294275e+06	1.349561e+06	NaN	1.511763e+09	NaN	NaN	1.100000e+01
50%	1.786180e+05	2.576360e+06	2.725426e+06	NaN	1.511966e+09	NaN	NaN	1.600000e+01
75%	2.255700e+05	3.860809e+06	4.145813e+06	NaN	1.512181e+09	NaN	NaN	2.000000e+01
max	1.018011e+06	5.163067e+06	5.161669e+06	NaN	1.512317e+09	NaN	NaN	2.300000e+01

3.3 构建模型和可视化

3.4.1 AARRR 模型进行用户行为转化

AARRR 模型因其掠夺式的增长方式也被称为海盗模型，是 Dave McClure 2007 提出的，核心就是 AARRR 漏斗模型，对应客户生命周期.

AARRR 分别代表了五个单词，又分别对应了产品生命周期中的五个阶段：

获取（Acquisition）：用户如何发现（并来到）你的产品？

激活（Activation）：用户的第一次使用体验如何？

留存（Retention）：用户是否还会回到产品（重复使用）？

收入（Revenue）：产品怎样（通过用户）赚钱？

传播（Refer）：用户是否愿意告诉其他用户？

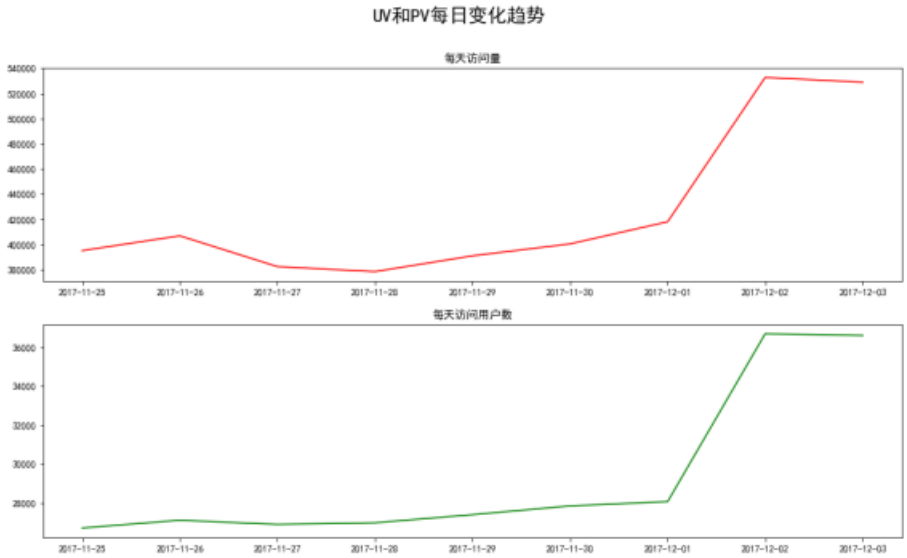
1. 对每天的用户行为 pv（日访问量）和 uv（日用户访问数）进行可视化

```
In [ ]: #AARRR模型
        #按日统计流量指标
        pv_daily = data_user.groupby('date').count()['UserID']
        pv_daily.head()

In [ ]: uv_daily = data_user.groupby('date')['UserID'].apply(lambda x: x.drop_duplicates().count())
        uv_daily.head()

In [ ]: pv_uv_daily = pd.concat([pv_daily,uv_daily], axis=1)
        pv_uv_daily.columns=['pv','uv']
        pv_uv_daily

In [ ]: pv_sum = pv_uv_daily['pv'].sum()
        uv_sum = pv_uv_daily['uv'].sum()
        pv_uv = pv_sum / uv_sum
        print(pv_sum, uv_sum, pv_uv)

In [26]: pv_uv_daily.corr(method='spearman')
```

Out[26]:

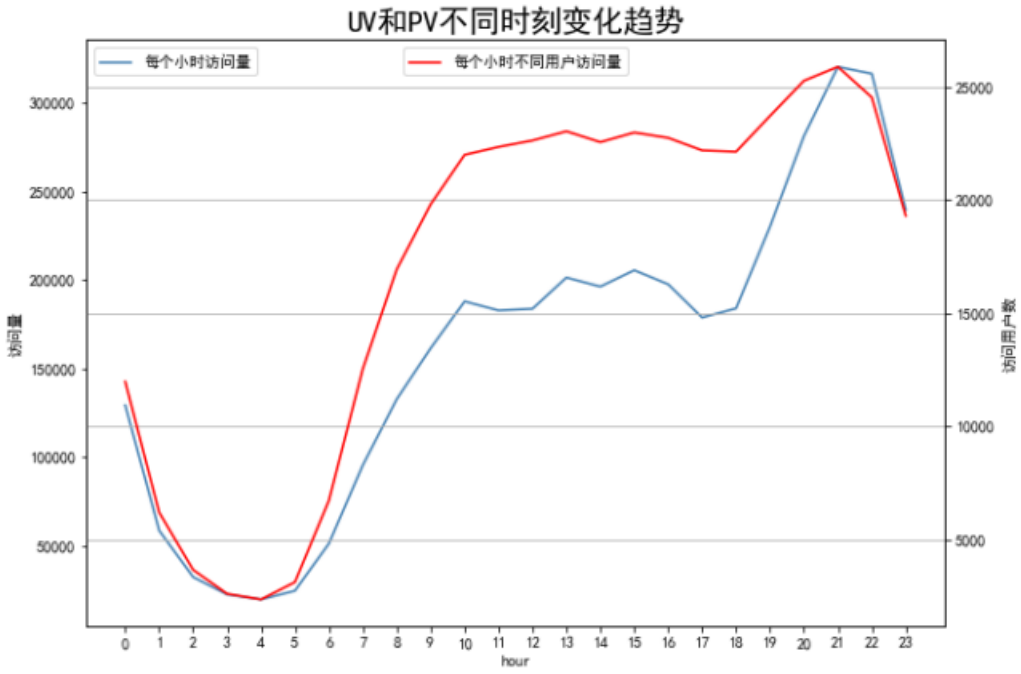
	pv	uv
pv	1.000000	0.816667
uv	0.816667	1.000000

2. 绘制 pv（日访问量）和 uv（日用户访问数）变化趋势图



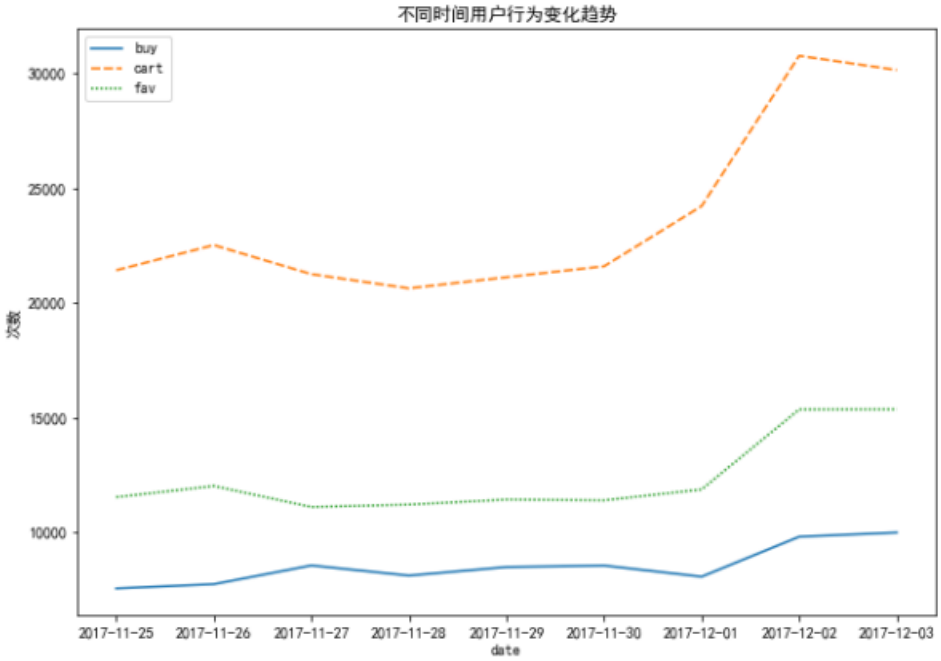
由上图发现工作日维持在低值，其中周二（11-27）的访问量达到统计范围内最低值；而 11 月 25 日、11 月 26 日和 12 月 2 日、12 月 3 日同为周末，但后者却有更多的活跃用户，环比增长率约为 32%，推测可能是平台做促销活动。检索可知正值“双十二”前夕，各类预热活动促进用户访问增长

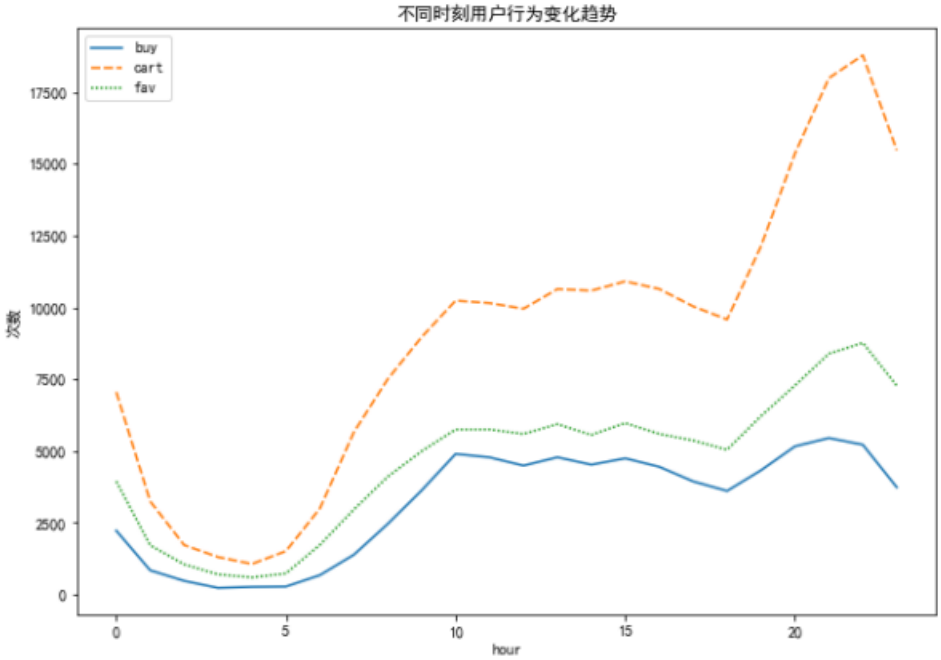
3. 绘制同一日内 pv 和 uv 不同时刻变化趋势图



结合人们日常作息规律，0 点至 6 点是休息时间，点击量处于低谷阶段；6 点至 10 点，人们慢慢开始工作，点击量开始回暖；10 点至 18 点为正常工作时间，点击量保持平稳；18 点至 20 点，人们相继下班休息，点击量不断升高；在 21 点至 22 点期间，点击量到达高峰。高峰期用户最活跃，建议商家在用户该时段，经常更新产品信息，黄金展位，活动推荐商品等

4. 不同时期用户行为变化趋势





5. 跳失率计算

跳失率 = 只浏览一个页面就离开的访问次数 / 该页面的全部访问次数

结果显示只有点击行为没有收藏、加购物车以及购买行为的总用户数是2196,除以总用户数37376得到跳失率为5.88%。说明用户对商品详情页的关注很大，商品详情页的商品描述，细节等吸引点不足，是流失用户的的重要原因之一。

4.1.2跳失率计算 跳失率 = 只浏览一个页面就离开的访问次数 / 该页面的全部访问次数

In [41]:
```
#跳失率 = 只浏览一个页面就离开的访问次数 / 该页面的全部访问次数
#结果显示只有点击行为没有收藏、加购物车以及购买行为的总用户数是2196，除以总用户数37376得到跳失率为5.88%。

pv_count = data_user[data_user.BehaviorType == 'pv'][['UserID']]
cart_count = data_user[data_user.BehaviorType == 'cart'][['UserID']]
fav_count = data_user[data_user.BehaviorType == 'fav'][['UserID']]
buy_count = data_user[data_user.BehaviorType == 'buy'][['UserID']]
cart_fav_buy_count = pd.concat([cart_count, fav_count, buy_count], axis=0)
```

In [42]:
```
#有收藏、加购物车以及购买行为的用户数是35180，则37376-35180=2196
cart_fav_buy_count.drop_duplicates(inplace=False)
```

Out[42]:

	UserID
0	173502
6	241235
7	131765
11	246648
21	21186
...	...
3827757	1017100
3828283	217083
3830799	240582
3830815	253744
3831654	15442

35180 rows × 1 columns

6. 用户留存率

留存用户：在某段时间开始使用产品，经过一段时间后仍然继续使用产品的用户，即为留存用户。

留存率=仍旧使用产品的用户量/最初的总用户量。

根据时间维度进行分类，留存率经常分为次日留存、3日留

存、7 日留存以及 30 日留存等。

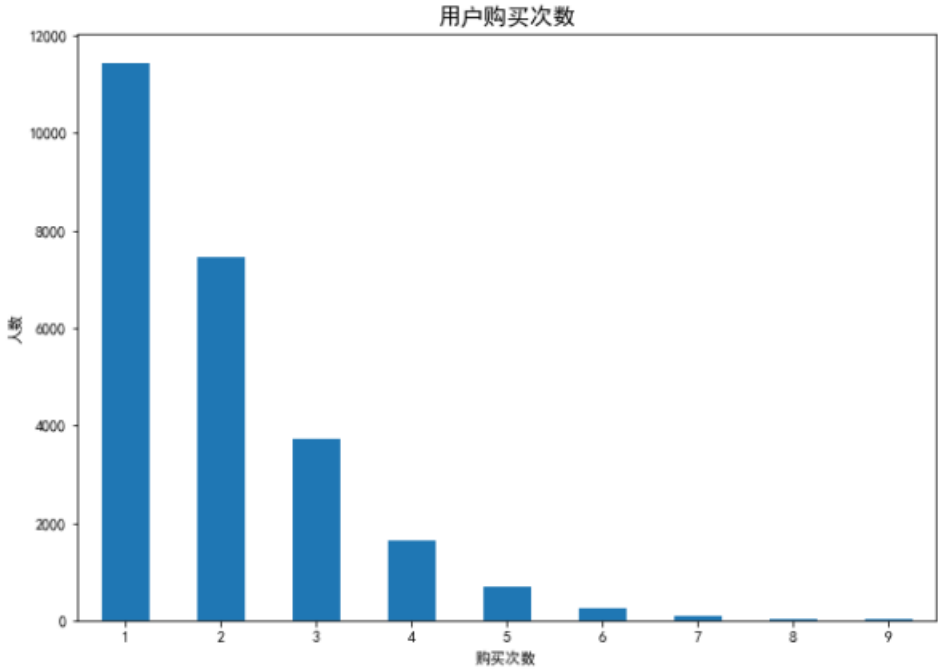
```
from datetime import timedelta

#建立n日留存率计算函数
def cal_retention(data,n): #n为n日留存
    user=[]
    date=pd.Series(data.date.unique()).sort_values()[:-n] #时间截取至最后一天的前n天
    retention_rates=[]
    new_users=[]
    retention_user=[]
    for i in date:
        new_user=set(data[data.date==i].UserID.unique())-set(user) #识别新用户，本案例中设初始用户量为零
        user.extend(new_user)  #将新用户加入用户群中
        #第n天留存情况
        user_nday=data[data.date==i+timedelta(n)].UserID.unique() #第n天登录的用户情况
        a=0
        for UserID in user_nday:
            if UserID in new_user:
                a+=1
        b = len(new_user)
        retention_rate=a/b #计算该天第n日留存率
        retention_rates.append(retention_rate) #汇总n日留存数据
        new_users.append(b) #汇总n日的新用户数
        retention_user.append(a) #汇总n日留存的用户数
    data_new_user = pd.Series(new_users, index=date)
    data_retention_user = pd.Series(retention_user, index=date)
    data_retention_rate = pd.Series(retention_rates,index=date)
    data_retention = pd.concat([data_new_user,data_retention_user,data_retention_rate], axis=1)
    data_retention.columns=['new_user','retention_user','retention_rate']
    return data_retention

data_retention1=cal_retention(data_user,1)
data_retention2=cal_retention(data_user,2)
data_retention6=cal_retention(data_user,6)
```

	新活跃人数	次日活跃人数	次日留存率	3日留存人数	3日留存率	7日留存人数	7日留存率
2017-11-25	26710	21071	0.788881	0.769861	0.769861	20711.0	0.775402
2017-11-26	6036	3904	0.646786	0.647946	0.647946	5906.0	0.978463
2017-11-27	2426	1464	0.603462	0.610058	0.610058	2347.0	0.967436
2017-11-28	1166	709	0.608062	0.652659	0.652659	NaN	NaN
2017-11-29	683	476	0.696925	0.691069	0.691069	NaN	NaN
2017-11-30	346	330	0.953757	0.959538	0.959538	NaN	NaN
2017-12-01	7	7	1.000000	1.000000	1.000000	NaN	NaN
2017-12-02	2	2	1.000000	NaN	NaN	NaN	NaN

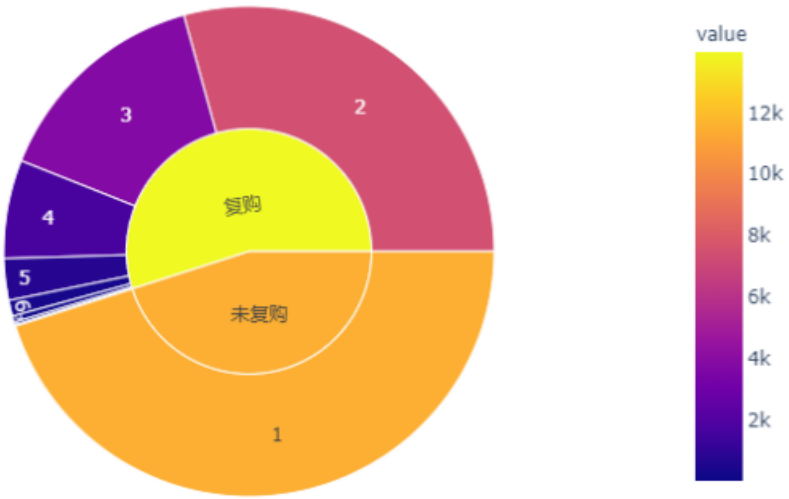
7. 复购率

绘制复购次数图

In [63]:
```
plt.figure(figsize=(10,7))
df_data_rebuy2.plot.bar()
plt.xticks(rotation=0)
plt.title('用户购买次数',fontsize=15)
plt.xlabel('购买次数')
plt.ylabel('人数')
plt.savefig('用户购买次数')
```



绘制饼状图



	复购情况	购买次数	value
0	未复购	1	11445
1	复购	2	7439
2	复购	3	3730
3	复购	4	1648
4	复购	5	698
5	复购	6	265
6	复购	7	104
7	复购	8	47
8	复购	9	24

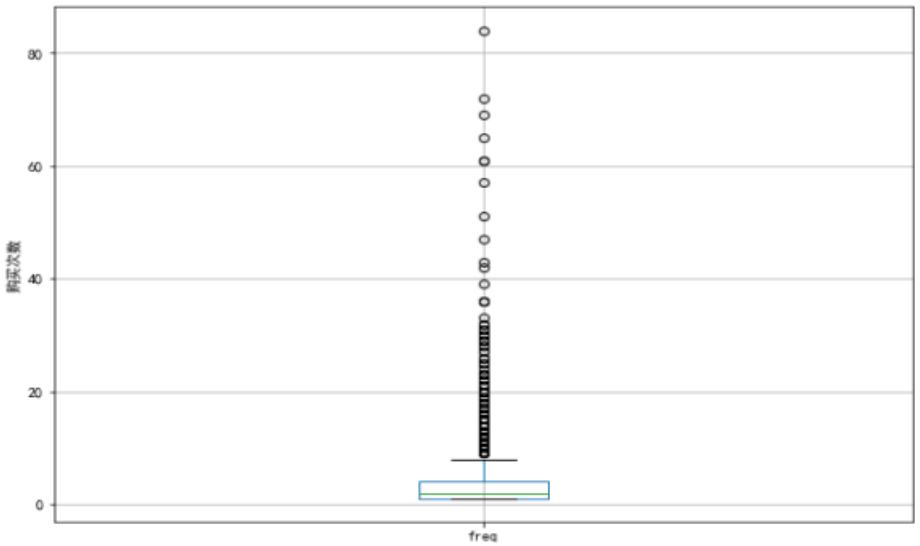
复购率计算以及购买次数分布

In [68]:
```
#复购率
data_rebuy[data_rebuy>=2].count()/data_rebuy.count()
```

Out[68]: 0.5494094488188976

```
buy_freq.describe()['freq']
```

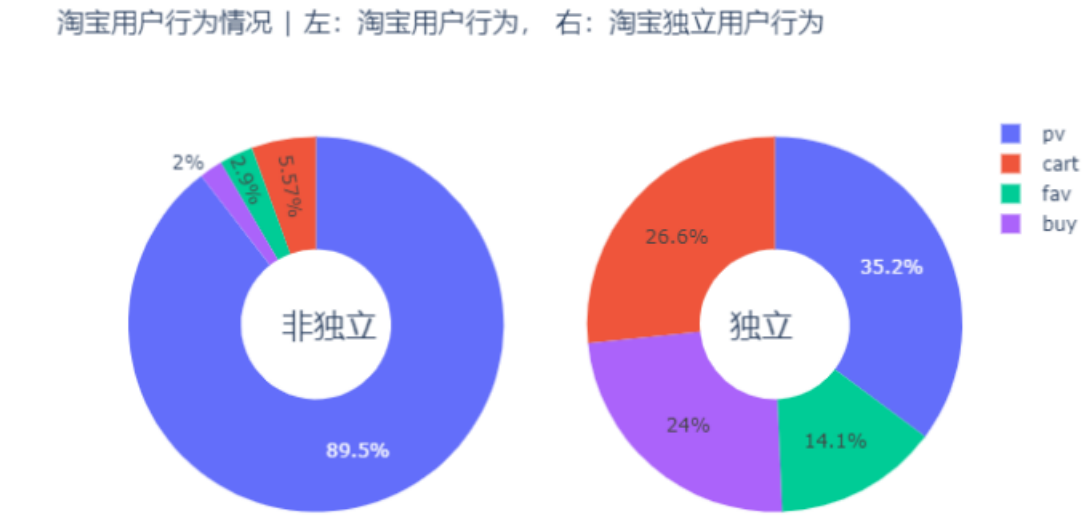
```
count    25400.000000
mean         3.019961
std          3.040463
min          1.000000
25%          1.000000
50%          2.000000
75%          4.000000
max         84.000000
Name: freq, dtype: float64
```



淘宝平台和用户的粘性很高，9 日内的复购率达到 54.94%。但有的用户购买次数高达到 84 次。9 天里有 84 次的购买行为，平均一天有 9 次购买行为，这不符合常理，因此可推断存在刷单现象。

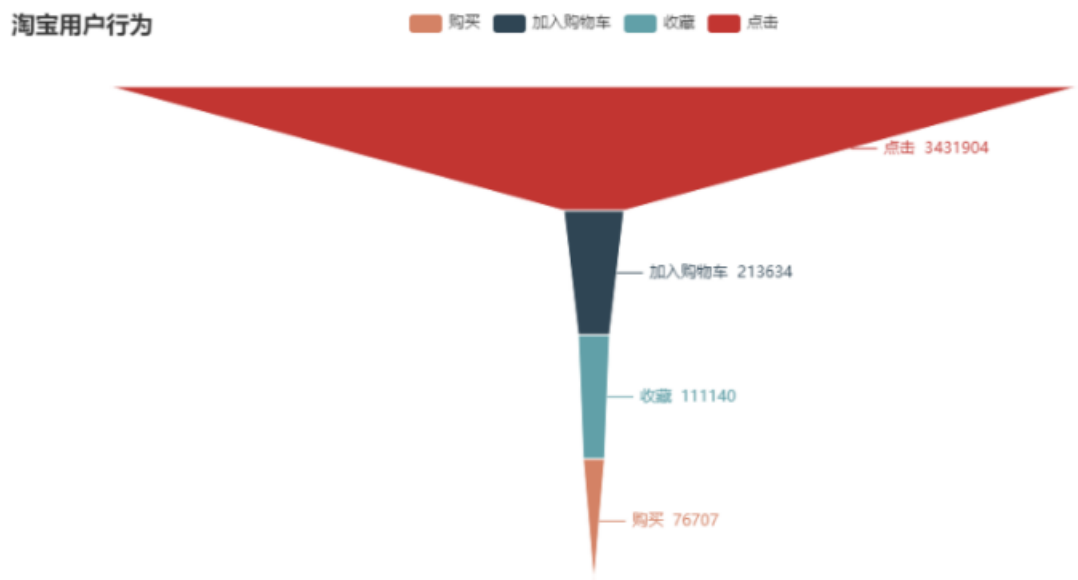
8. 用户行为分析

```
# 多子图绘制 如：将上面用到的图形一起绘制
# 导入subplots（类似matplotlib）
from plotly.subplots import make_subplots

labels = df_userbehavior['behavior']

# Create subplots: use 'domain' type for Pie subplot
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=labels, values=data_user_count.values, name="淘宝用户行为"),
              1, 1)
fig.add_trace(go.Pie(labels=labels, values=df_userbehavior['count'], name="淘宝独立用户行为"),
              1, 2)

# Use `hole` to create a donut-like pie chart
fig.update_traces(hole=.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="淘宝用户行为情况 | 左：淘宝用户行为， 右：淘宝独立用户行为",
    # Add annotations in the center of the donut pies.
    annotations=[dict(text='非独立', x=0.18, y=0.5, font_size=20, showarrow=False),
                 dict(text='独立', x=0.8, y=0.5, font_size=20, showarrow=False)])
fig.show()
```



用户点击行为占总行为数的89.5%，收藏和加购行为加起来的行为数只占总行为数的8.47%，而对于独立用户来说，点击行为的占比明显缩小为35.2%。推测用户可能在挑选产品环节浪费了较多的时间。

用户行为分析的漏斗模型



项目	流失率
pv_to_cart	93.78%
cart_to_fav	47.98%
fav_to_buy	30.98%
pv_to_buy	97.76%

从点击到购买的全过程中，流失率主要集中在点击到加入购物车这一环节，流失率高达 93.78%，收藏及加入购物车后购买商品的可能性增大。

3.4.2 RFM 模型进行用户价值分析

RFM 模型概述

RFM 模型是网点衡量当前用户价值和客户潜在价值的重要工具和手段。RFM 是 Rencency（最近一次消费），Frequency（消费频率）、Monetary（消费金额），三个指标首字母组合，如图所示

因为本数据集没有提供 M（消费金额）列，因此只能通过 R（最近一次购买时间）和 F（消费频率）的数据对客户价值进行打分

RFM	业务含义	1分	2分
R	最近交易日期与2017.12.4距离天数	3~9	0~3
F	购买次数	0~2	2~84

其中，

RF=(1,1)为重要挽回客户；

RF=(1,2)为重要唤回客户；

RF=(2,1)为重要深耕客户：

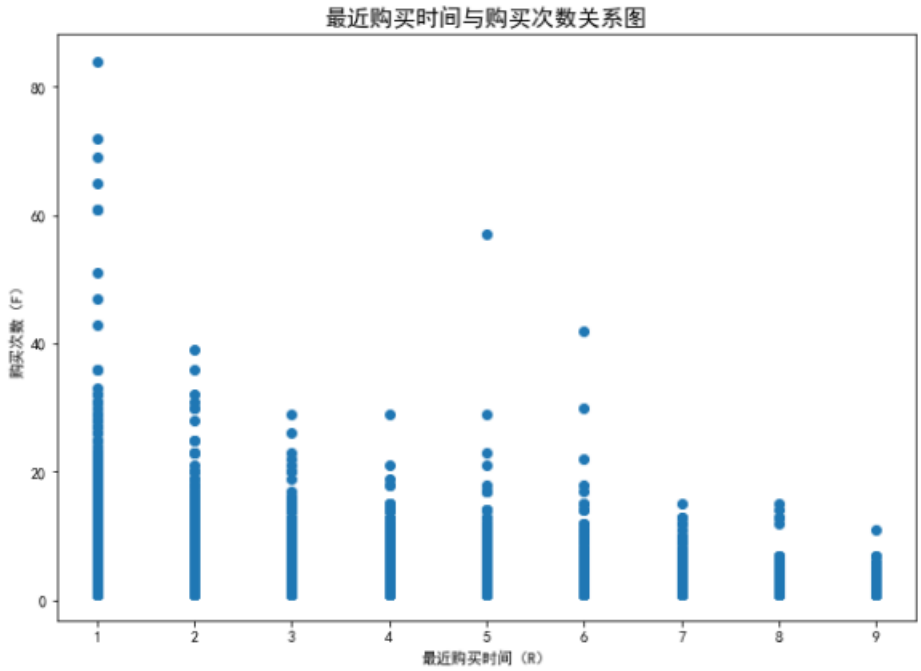
RF=(2,2)为重要价值客户。

4.2.4绘制用户重要度的树状图

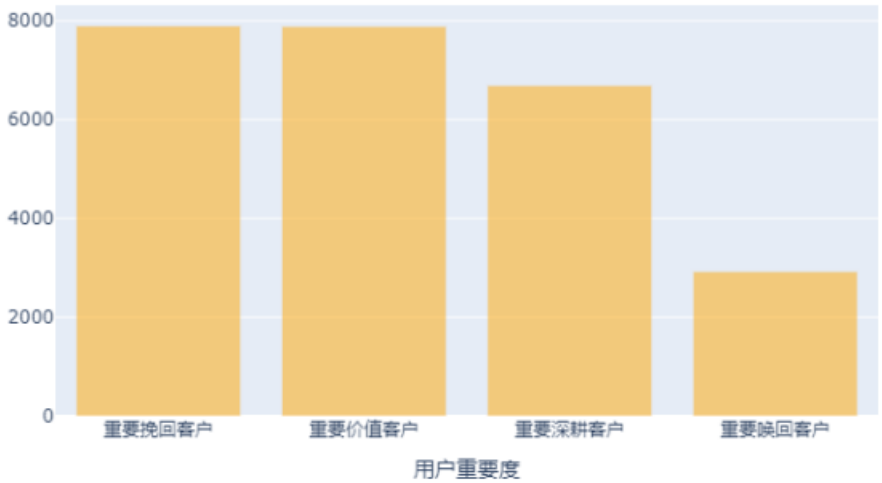
```
trace_basic = [go.Bar(x = rfm['rank'].value_counts().index,
                      y = rfm['rank'].value_counts().values,
                      marker = dict(color='orange'), opacity=0.50)]
layout = go.Layout(title='用户等级情况', xaxis=dict(title='用户重要度'))
figure_basic = go.Figure(data=trace_basic, layout=layout)
figure_basic
```

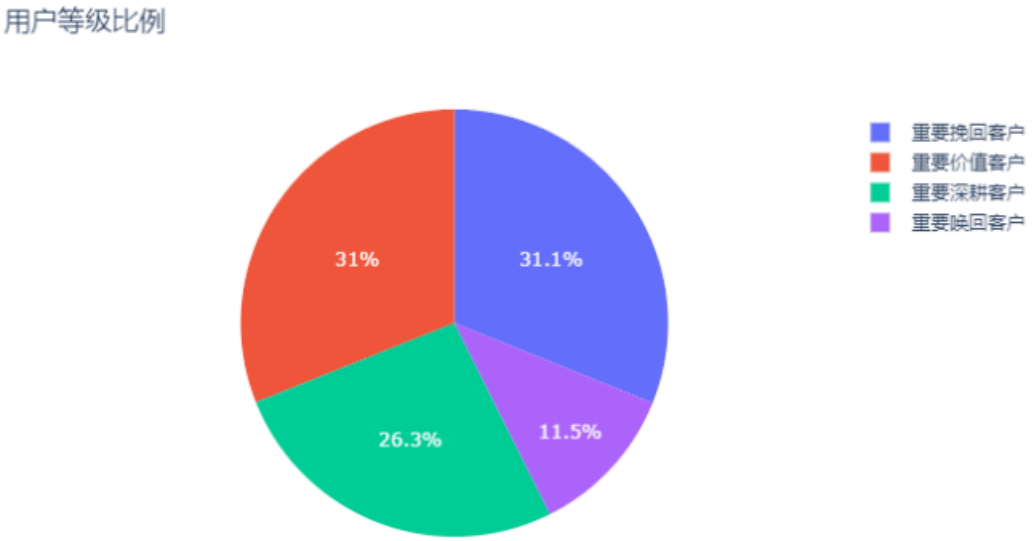
4.2.5绘制用户等级比例饼状图

```
]: trace = [go.Pie(labels=rfm['rank'].value_counts().index,
                   values = rfm['rank'].value_counts().values,
                  textfont = dict(size=12,color='white'))]
layout = go.Layout(title='用户等级比例')
figure_pie = go.Figure(data=trace, layout=layout)
figure_pie
```



用户等级情况





四 结论及建议

本项目基于 AARRR 模型，从五个维度研究了关于淘宝用户行为分析的问题。

成功实现以下成果：

1. 通过对日访问量和单日用户访问数以及用户行为变化趋势的分析，提出了增加平台流量的三种方法

2. 通过跳失率计算和用户行为分析，成功推测出用户流失大的环节，提出了增加加购转化率的建议

3. 通过对留存率进行时间维度的分类，提出将购物活动划分时间段的方法。

4.通过研究用户复购情况,提出了提高商家收入的三条建议

5.最后结合自己的思考,提出了平台建立对产品的质量监控机制的建议

本项目基于RFM模型，对用户价值进行了分层

成功实现以下成果:

1. 得到了不同类型的客户群体，找出最具价值的核心付费用户群。

2.针对不同的客户群体，采取了不同的激励方案:

对于RF=22的重要价值客户，应该提高满意度，增加留存。

对于RF=21的重要深耕客户，可通过折扣或捆绑销售等活动，提高购买频率。

对于RF=12的重要唤回客户,分析其偏好,更精准地推送商品,以防流失。

对于RF=11的重要挽回客户，可考虑发放限时优惠券，促进关注与消费。