

```
In [1]:

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

In [2]:

user=pd.read_csv('/home/kesci/input/89031322/tianchi_fresh_comp_train_user.csv')
```

```
In [3]:

user.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15463110 entries, 0 to 15463109
Data columns (total 6 columns):
user_id          int64
item_id          int64
behavior_type    int64
user_geohash     object
item_category    int64
time            object
dtypes: int64(4), object(2)
memory usage: 707.8+ MB
```

```
In [4]:

# 快速查看统计信息
user.describe()
```

	user_id	item_id	behavior_type	item_category
count	1.546311e+07	1.546311e+07	1.546311e+07	1.546311e+07
mean	7.015207e+07	2.023169e+08	1.153780e+00	6.827181e+03
std	4.572019e+07	1.167524e+08	5.440445e-01	3.810410e+03
min	4.920000e+02	3.700000e+01	1.000000e+00	2.000000e+00
25%	3.021406e+07	1.014015e+08	1.000000e+00	3.687000e+03
50%	5.638708e+07	2.022669e+08	1.000000e+00	6.054000e+03
75%	1.166482e+08	3.035247e+08	1.000000e+00	1.025800e+04
max	1.424430e+08	4.045625e+08	4.000000e+00	1.408000e+04

```
In [5]:

user.isnull().sum()

user_id          0
item_id          0
behavior_type    0
user_geohash     8207386
item_category    0
time            0
dtype: int64
```

1) 删除重复值

```
In [6]:

user.drop_duplicates(keep='last',inplace=True)
```

2) 将time转换为datetime格式

```
In [7]:

user['time']=pd.to_datetime(user['time'])
```

3) 提取出日期和时间

```
In [8]:  
  
user['dates'] = user.time.dt.date  
user['month'] = user.dates.values.astype('datetime64[M]')  
user['hours'] = user.time.dt.hour
```

#### 4) 转换数据类型

```
In [9]:  
  
user['behavior_type']=user['behavior_type'].apply(str)
```

```
In [10]:  
  
user['user_id']=user['user_id'].apply(str)
```

```
In [11]:  
  
user['item_id']=user['item_id'].apply(str)
```

#### 1) 统计每日PV和UV数据

```
In [12]:  
  
pv_day=user[user.behavior_type=="1"].groupby("dates")["behavior_type"].count()
```

```
In [13]:  
  
uv_day=user[user.behavior_type=="1"].drop_duplicates(["user_id","dates"]).group
```

#### 2) 分析每天的pv与uv的趋势

#### 3) pv、uv差异分析 (by day)

```
In [15]:

pv_uv = pd.concat([pv_day, uv_day], join='outer', axis=1)
pv_uv.columns = ['pv_day', 'uv_day']

new_day=pv_uv.diff()
new_day.columns=['new_pv', 'new_uv']
new_day
```

	new_pv	new_uv
dates		
2014-11-18	NaN	NaN
2014-11-19	9121.0	61.0
2014-11-20	-12504.0	-71.0
2014-11-21	-20029.0	-228.0
2014-11-22	18068.0	-122.0
2014-11-23	37567.0	432.0
2014-11-24	-7484.0	140.0
2014-11-25	-17090.0	-178.0
2014-11-26	-6583.0	-138.0
2014-11-27	7738.0	-61.0
2014-11-28	-21030.0	-208.0
2014-11-29	14585.0	4.0
2014-11-30	40563.0	342.0
2014-12-01	-3456.0	300.0
2014-12-02	5281.0	-7.0
2014-12-03	17074.0	157.0
2014-12-04	-25985.0	-111.0
2014-12-05	-30945.0	-237.0
2014-12-06	54759.0	37.0
2014-12-07	22581.0	291.0
2014-12-08	3030.0	30.0
2014-12-09	13630.0	-10.0
2014-12-10	15549.0	117.0
2014-12-11	101816.0	694.0
2014-12-12	220781.0	1542.0
2014-12-13	-338647.0	-1988.0
2014-12-14	6078.0	-130.0
2014-12-15	-11345.0	238.0
2014-12-16	-12248.0	-166.0
2014-12-17	-8157.0	-218.0
2014-12-18	-9338.0	-77.0

In [16]:

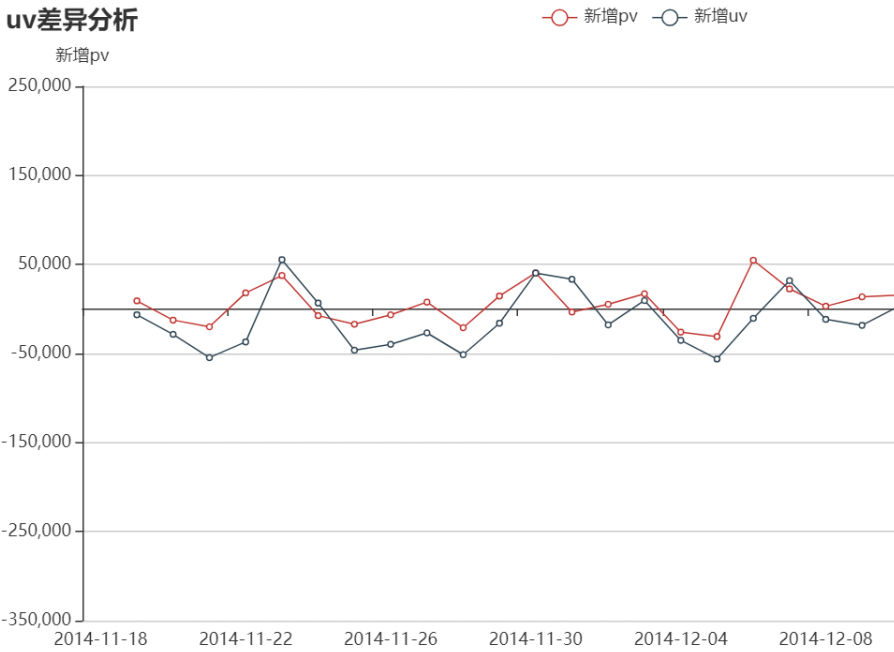
```

attr = new_day.index
v = new_day.new_uv
w = new_day.new_pv

li=(
    Line(init_opts=opts.InitOpts(width="1000px",height="500px"))
    .add_xaxis(xaxis_data=attr)
    .add_yaxis(
        "新增pv",
        w,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .extend_axis(
        yaxis=opts.AxisOpts(
            name="新增uv",
            type_="value",
            min_=-2000,
            max_=1600,
            interval=400,
            axislabel_opts=opts.LabelOpts(formatter="{value}"),
        )
    )
    .set_global_opts(
        tooltip_opts=opts.TooltipOpts(
            is_show=True, trigger="axis", axis_pointer_type="cross"
        ),
        xaxis_opts=opts.AxisOpts(
            type_="category",
            axispointer_opts=opts.AxisPointerOpts(is_show=True, type_="shadow")
        ),
        yaxis_opts=opts.AxisOpts(
            name="新增pv",
            type_="value",
            min_=-350000,
            max_=250000,
            interval=100000,
            axislabel_opts=opts.LabelOpts(formatter="{value}"),
            axistick_opts=opts.AxisTickOpts(is_show=True),
            splitline_opts=opts.SplitLineOpts(is_show=True),
        ),
        title_opts=opts.TitleOpts(title="pv、uv差异分析"),
    )
)
il=(
    Line()
    .add_xaxis(xaxis_data=attr)
    .add_yaxis("新增uv",v,yaxis_index='1',label_opts=opts.LabelOpts(is_show=False)
)
c=li.overlap(il)
c.render_notebook()

```

pv、uv差异分析



4) 不同时期用户行为分析

In [17]:

```

shopping_cart= user[user.behavior_type == '3'].groupby('dates')['behavior_type']
collect=user[user.behavior_type=='2'].groupby('dates')['behavior_type'].count()
buy=user[user.behavior_type=='4'].groupby('dates')['behavior_type'].count()

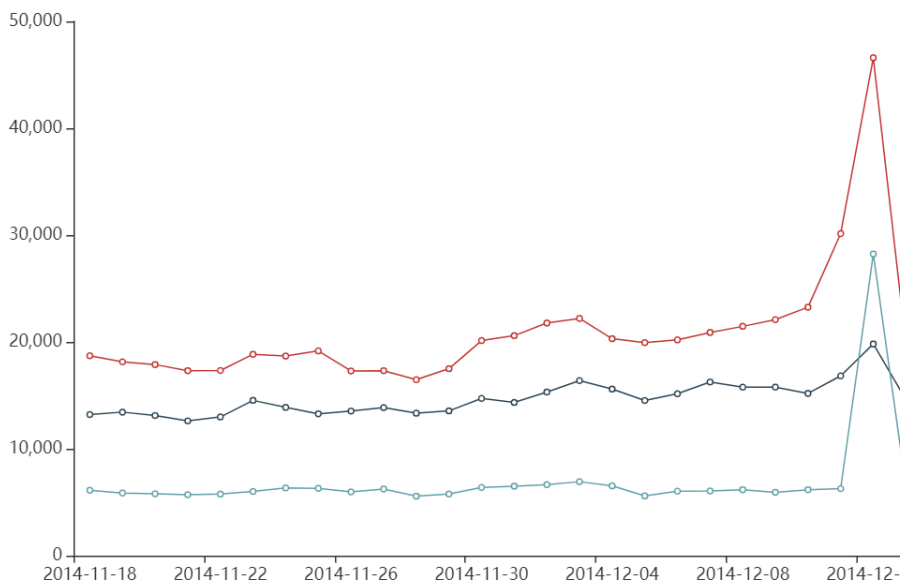
attr_a=list(shopping_cart.index)
v_1=shopping_cart.values.tolist()
v_2=collect.values.tolist()
v_3=buy.values.tolist()

b=(
    Line()
    .add_xaxis(xaxis_data=attr_a)
    .add_yaxis(
        "加购人数",
        v_1,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .add_yaxis(
        "收藏人数",
        v_2,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .add_yaxis(
        "购买人数",
        v_3,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="不同时期用户行为数据"))
)
b.render_notebook()

```

不同时期用户行为数据

—○— 加购人数 —○— 收藏人数 —○— 购买人数



## 1) 把dates列转换为datetime类型

In [18]:

```
user['dates']=pd.to_datetime(user['dates'])
```

## 2) 选取活动数据子集和日常数据子集

In [19]:

```
active=user[user["dates"].isin(["2014/12/11","2014/12/12","2014/12/13"])]
```

In [20]:

```
daily=user[user["dates"].isin(["2014/12/11","2014/12/12","2014/12/13"])]
```

### 3) 活动期间不同时段的用户行为分析

In [21]:

```

from pyecharts.charts import Bar
# 活动数据
cart_h= active[active.behavior_type == '3'].groupby('hours')['behavior_type'].c
collect_h=active[active.behavior_type=='2'].groupby('hours')['behavior_type'].c
buy_h=active[active.behavior_type=='4'].groupby('hours')['behavior_type'].count
uv_h=active[active.behavior_type== '1'].groupby('hours')['user_id'].count()

attr_h=list(cart_h.index)
h1=np.around(cart_h.values/3,decimals=0).tolist()
h2=np.around(collect_h.values/3,decimals=0).tolist()
h3=np.around(buy_h.values/3,decimals=0).tolist()
h4=np.around(uv_h.values/3,decimals=0).tolist()

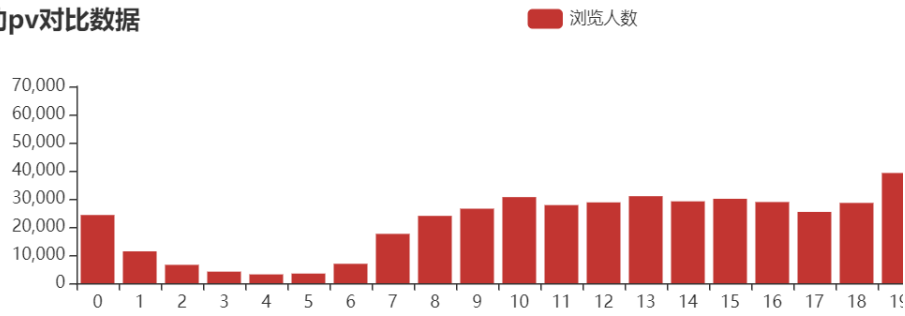
h=(
    Line(init_opts=opts.InitOpts(width="1000px",height="500px"))
    .add_xaxis(xaxis_data=attr_h)
    .add_yaxis(
        "加购人数",
        h1,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .add_yaxis(
        "收藏人数",
        h2,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .add_yaxis(
        "购买人数",
        h3,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .set_global_opts(
        xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate=15)),
        title_opts=opts.TitleOpts(title="日均各时段活动用户行为",pos_top="48%"),
        legend_opts=opts.LegendOpts(pos_top="48%"),
    )
)
bar=(
    Bar()
    .add_xaxis(xaxis_data=attr_h)
    .add_yaxis(
        "浏览人数",
        h4,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="活动pv对比数据"),
    )
)

ggrid = (
    Grid()
    .add(bar, grid_opts=opts.GridOpts(pos_bottom="60%"))
    .add(h, grid_opts=opts.GridOpts(pos_top="60%"))
)
ggrid.render_notebook()

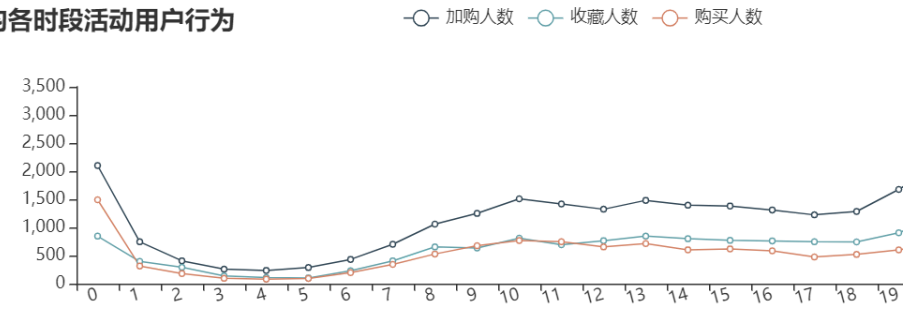
```



活动pv对比数据



日均各时段活动用户行为



大促集中在0点，因此用户的购买高峰也出现在0点。  
高峰集中在21点到22点之间，因此可以在20点前，改好促销页面，吸引顾客参加0点的活动。

4) 日常期间不同时段的用户行为分析

In [22]:

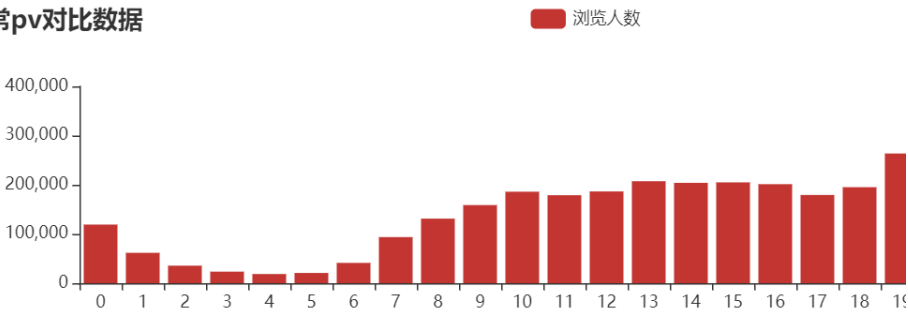
```
# 日常数据
cart_d= daily[daily.behavior_type == '3'].groupby('hours')['behavior_type'].count()
collect_d=daily[daily.behavior_type=='2'].groupby('hours')['behavior_type'].count()
buy_d=daily[daily.behavior_type=='4'].groupby('hours')['behavior_type'].count()
uv_d=daily[daily.behavior_type=='1'].groupby('hours')['user_id'].count()

attr_d=list(cart_d.index)
d1=np.around(cart_d.values/28,decimals=0).tolist()
d2=np.around(collect_d.values/28,decimals=0).tolist()
d3=np.around(buy_d.values/28,decimals=0).tolist()
d4=np.around(uv_d.values/3,decimals=0).tolist()

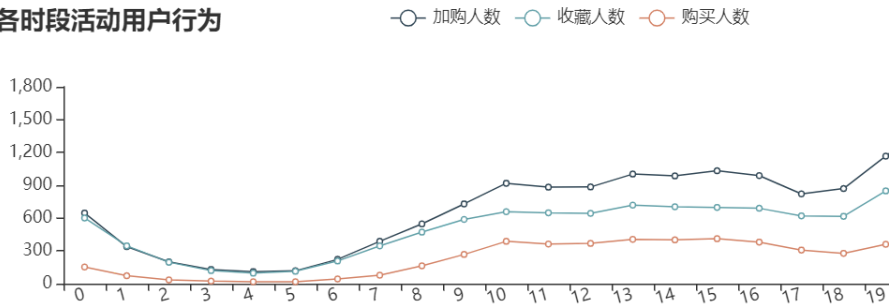
d=(
    Line(init_opts=opts.InitOpts(width="1000px",height="500px"))
    .add_xaxis(xaxis_data=attr_d)
    .add_yaxis(
        "加购人数",
        d1,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .add_yaxis(
        "收藏人数",
        d2,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .add_yaxis(
        "购买人数",
        d3,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .set_global_opts(
        xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate=15)),
        title_opts=opts.TitleOpts(title="日均各时段活动用户行为",pos_top="48%"),
        legend_opts=opts.LegendOpts(pos_top="48%"),
    )
)
y=(
    Bar()
    .add_xaxis(xaxis_data=attr_d)
    .add_yaxis(
        "浏览人数",
        d4,
        label_opts=opts.LabelOpts(is_show=False)
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="日常pv对比数据"),
    )
)

ggrid = (
    Grid()
    .add(y, grid_opts=opts.GridOpts(pos_bottom="60%"))
    .add(d, grid_opts=opts.GridOpts(pos_top="60%"))
)
ggrid.render_notebook()
```

日常pv对比数据

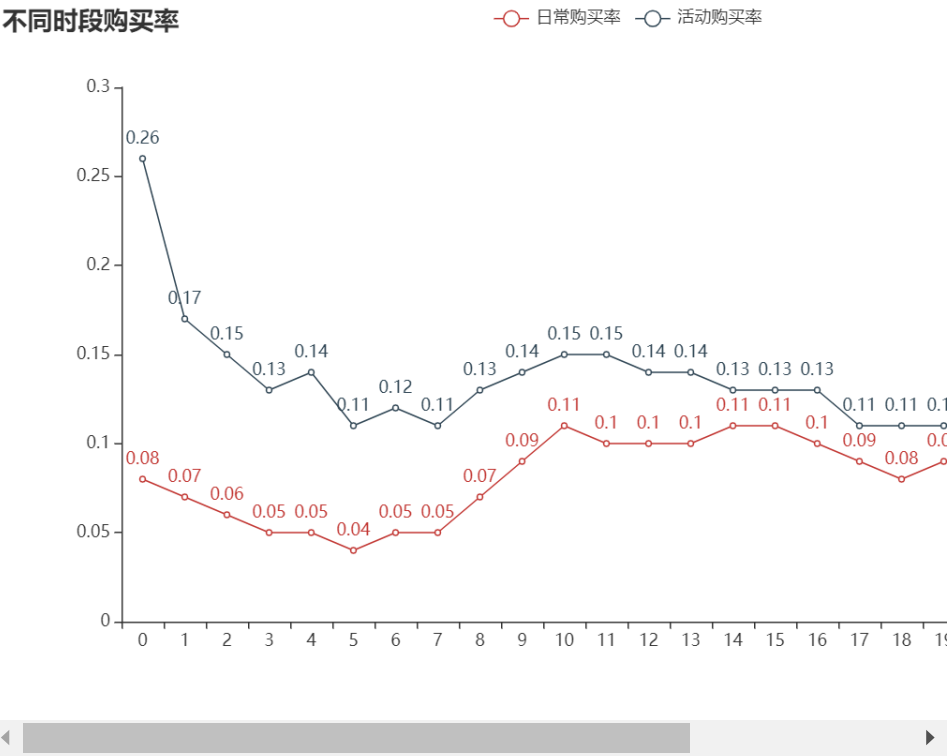


日均各时段活动用户行为



日常期间，pv、加购、收藏的高峰出现在晚上21点到22点之间；浏览高峰也是集中在晚上21点到22点之间。可以集中在这个时段进行促销活动。

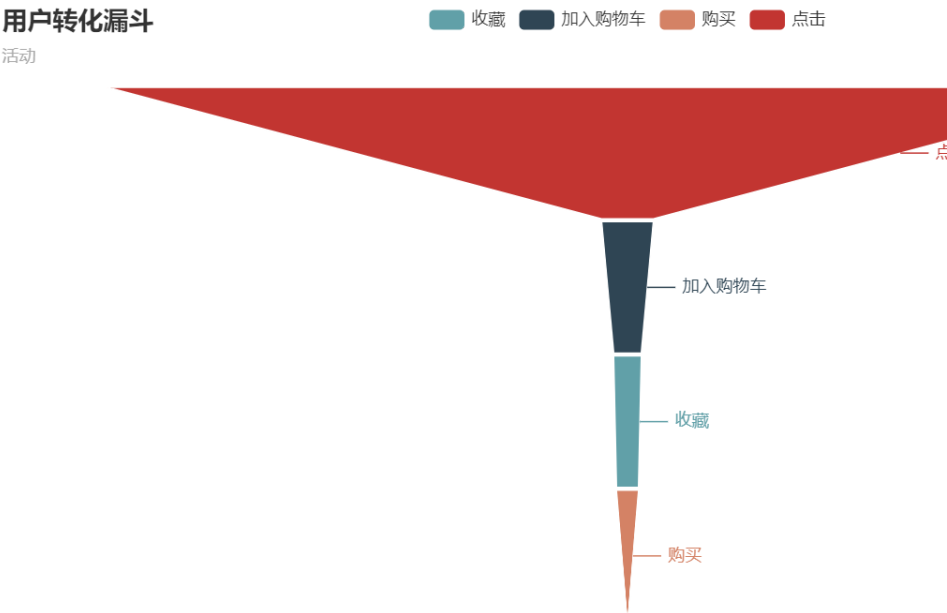
```
In [23]:  
  
# 活动时购买率  
hour_buy_user_num = active[active.behavior_type == '4'].drop_duplicates(['user_  
hour_active_user_num = active.drop_duplicates(['user_id', 'dates', 'hours']).grc  
hour_buy_rate = hour_buy_user_num / hour_active_user_num  
attr_o = list(hour_buy_user_num.index)  
vo_2 =np.around(hour_buy_rate.values,decimals=2)  
# 日常时购买率  
hour_buy_daily_num = daily[daily.behavior_type == '4'].drop_duplicates(['user_i  
hour_active_daily_num = daily.drop_duplicates(['user_id', 'dates', 'hours']).grc  
daily_buy_rate = hour_buy_daily_num / hour_active_daily_num  
vi_2 =np.around(daily_buy_rate.values,decimals=2)  
  
hbu=(  
    Line()  
    .add_xaxis(xaxis_data=attr_o)  
    .add_yaxis(  
        "日常购买率",  
        vi_2,  
    )  
    .add_yaxis(  
        "活动购买率",  
        vo_2,  
    )  
    .set_global_opts(title_opts=opts.TitleOpts(title="不同时段购买率"))  
)  
  
hbu.render_notebook()
```



购买率高峰，也出现在晚上21点。  
因此可以考虑，这个时段增加吸引用户购买的措施。

1) 活动期间的转化漏斗

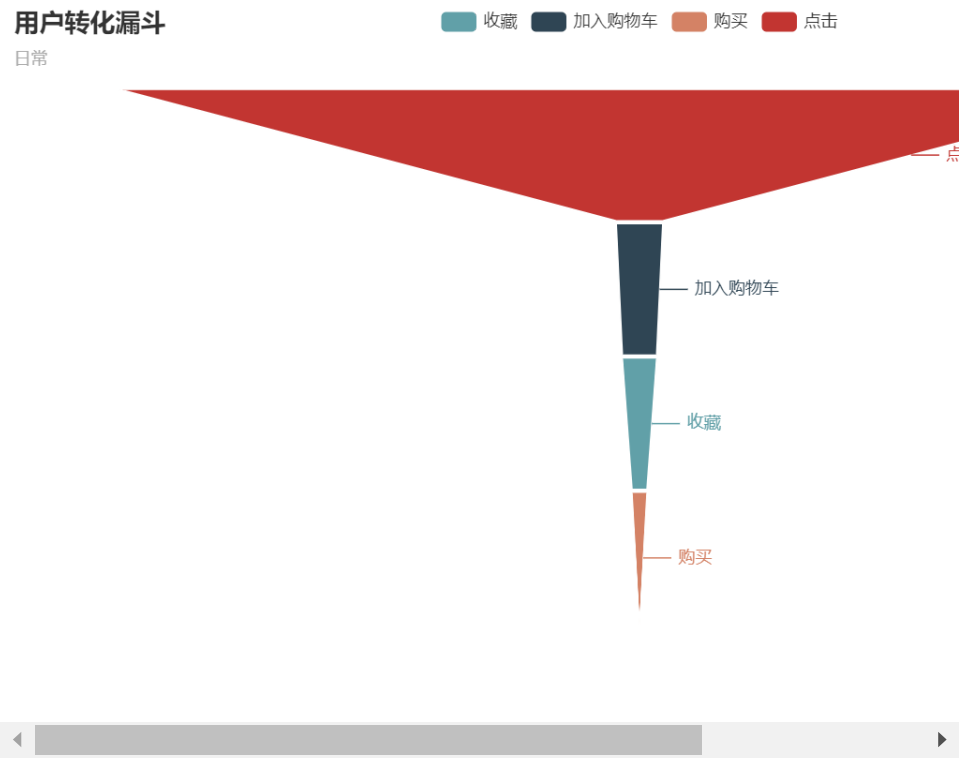
```
In [24]:  
  
from pyecharts.charts import Funnel  
  
# 活动转化  
a_pv=active[active.behavior_type=="1"]["user_id"].count()  
a_cart=active[active.behavior_type=="3"]["user_id"].count()  
a_collect=active[active.behavior_type=="2"]["user_id"].count()  
a_buy=active[active.behavior_type=="4"]["user_id"].count()  
  
a_attr=["点击","加入购物车","收藏","购买"]  
values=[np.around((a_pv/a_pv*100),2),  
        np.around((a_cart/a_pv*100),2),  
        np.around((a_collect/a_pv*100),2),  
        np.around((a_buy/a_pv*100),2),  
        ]  
  
data = [[a_attr[i], values[i]] for i in range(len(a_attr))]  
  
a=(  
    Funnel()  
    .add(  
        series_name="用户行为",  
        data_pair=data,  
        gap=2,  
        tooltip_opts=opts.TooltipOpts(trigger="item", formatter="{a} <br/>{b} :",  
        label_opts=opts.LabelOpts(is_show=True, position="outside"),  
        itemstyle_opts=opts.ItemStyleOpts(border_color="#fff", border_width=1),  
    )  
    .set_global_opts(title_opts=opts.TitleOpts(title="用户转化漏斗", subtitle="活  
动"))  
  
a.render_notebook()
```



从点击到加入购物车的转化率只有4.97%，购买的只有2%。  
说明点击浏览量不少，但是吸引不了顾客购买。  
因此，可以尝试提高加购率和收藏率。

2) 日常期间的转化漏斗

```
In [25]:  
  
# 日常转化  
l_pv=daily[daily.behavior_type=="1"]["user_id"].count()  
l_cart=daily[daily.behavior_type=="3"]["user_id"].count()  
l_collect=daily[daily.behavior_type=="2"]["user_id"].count()  
l_buy=daily[daily.behavior_type=="4"]["user_id"].count()  
  
l_attr=["点击","加入购物车","收藏","购买"]  
valuel=[np.around((l_pv/l_pv*100),2),  
        np.around((l_cart/l_pv*100),2),  
        np.around((l_collect/l_pv*100),2),  
        np.around((l_buy/l_pv*100),2),  
        ]  
  
datal = [[l_attr[i], valuel[i]] for i in range(len(l_attr))]  
  
dy=(  
    Funnel()  
    .add(  
        series_name="用户行为",  
        data_pair=datal,  
        gap=2,  
        tooltip_opts=opts.TooltipOpts(trigger="item", formatter="{a} <br/> {b} :",  
        label_opts=opts.LabelOpts(is_show=True, position="outside"),  
        itemstyle_opts=opts.ItemStyleOpts(border_color="#fff", border_width=1),  
    )  
    .set_global_opts(title_opts=opts.TitleOpts(title="用户转化漏斗", subtitle="日常"  
))  
dy.render_notebook()
```



总体点击量中，有4.45%加入购物车，有3.3%收藏，而到最后只有1.4%购买。  
购买的转化率最低。  
细分来看，“点击-加入购物车”这一环节的转化率最低。  
因此，我们可优化“点击-加入购物车”这一环节，增加优惠券环节，鼓励用户收藏、加购，并刺激用户的购买欲望。

——这份数据还能做用户地理分布的分析，用户对商品的偏好分析，本人懒，练习就到这里。