

# 电子商务网站用户分群

## 1 项目背景

随着计算机技术与通信技术的日益成熟和广泛应用，互联网自 90 年代初开始得到迅猛的发展，至今虽只有十几年时间，但已发展成为信息时代的人类生活中不可或缺的部分，成为全球范围内信息传播的主要渠道，与此同时，连入互联网的用户越来越多。目前，互联网的用户规模已不容小视，互联网市场潜力巨大，各大网站运营商都在采取积极的措施，分析用户的行为特征，对不同客户群提供差异化的服务，以达到精准营销的目的。比如，有些网站根据用户的注册资料，性别、年龄、区域、职业等信息对用户进行分群，但这种分群方式是“粗犷”的，未能考虑到用户的行为特征和兴趣偏好，分群结果难以为精准营销提供决策的支持。

网站运营商想要在浩如烟海的互联网用户中找到目标客户存在很大困难，具体表现为：运营商不知道目标用户浏览网站时有什么样的行为特征和习惯？运营商急需快速有效的方法解决用户分群问题。在这一背景下，本案例采用数据挖掘的手段分析用户过去的浏览行为，在此基础上建立用户自动分群模型。

本案例的研究对象是广州泰迪智能科技有限公司旗下的泰迪杯竞赛网站，泰迪智能科技有限公司是一家专门从事大数据挖掘研发、咨询和培训服务的高科技企业。泰迪杯竞赛网致力于为用户提供丰富的泰迪杯竞赛信息、数据挖掘培训咨询，并提供往届优秀作品作为参考，以及面向高校的丰富教学资源，如：案例教程、教学视频、教学书籍、建模工具等。访问用户的增加使得网站越发难以掌握用户的需求。为更好满足用户的需求，本案例依据用户的历史浏览记录，研究用户的兴趣偏好，分析需求并发现用户的兴趣点，从而将用户分成不同群体。公司后续可以针对不同群体提供差异化的服务，提高用户的使用体验。

## 2 项目目标

- (1) 依据用户的历史浏览记录，分析用户的行为特征和兴趣偏好。
- (2) 根据用户的行为特征和兴趣偏好将用户划分成不同的群体，分析各群体的属性特征。

## 3 项目步骤

### 3.1 工程前期准备

#### 3.1.1 导入数据

##### (1) 介绍数据

用户访问网站时，系统会自动记录用户访问网站的日志。该网站被访问的数据记录（部分字段），如表 3-1 所示。

表 3-1 用户访问记录表

id	page_path	Userid	seeeionid	ip	data_time
1	/zytj/index.j html	NA	DE80E709835F8AB1A381 96185B05FDBC	218.28.23.137	2016/7/14 18:33
2	/zytj/index.j html	NA	ED095CA37DB28D14041 24B4988CAFB9F	218.28.23.137	2016/7/14 18:33
3	/txxm/index. jhtml	8180	773F9B491EF1027B76698 C489DEB9DB9	188.165.225.224	2016/7/14 18:34
4	/notice/614.j html	NA	E32144406C1DEAB298F E4677846A449D	180.153.214.152	2016/7/14 18:35
5	/stpj/626.jht ml	8181	FBD4EB0F3E6390A49399 7B22B0DE51AD	180.153.206.20	2016/7/14 18:35
6	/thirdtipdm/i ndex.jhtml	NA	0430EF0B7E5CD8A3831 E78290DD2CED3	111.206.36.19	2016/7/14 18:35

表 3-1 记录了访问序号、内容 id、访问内容、用户 id、ip 等多项属性的记录，并针对其中各个属性进行了说明，如表 3-2 所示。

表 3-2 访问记录属性表

属性名称	属性说明	属性名称	属性说明
id	访问序号	browser_type	浏览器类型

content_id	内容 id	browser_version	浏览器版本
page_path	网址	platform_type	平台类型
username	用户名称	platform_series	平台系列
userid	用户 id	platform_version	平台版本
sessionid	一次浏览标识	data_time	访问时间
ip	ip 地址	mobile_type	手机类型
country	国家	agent	代理商
area	区域	uniqueVisitorId	唯一浏览 ID

## (2) 上传数据到 Python 数据挖掘建模平台

在新增数据源上，选择本地上传数据，如图 1 所示。



图 1 本地上传数据源

在本地路径上选择文件，填写在平台新建的目标表名，如图 2 所示。



图 2 本地选择文件上传

根据文件的数据，可以修改文件的字段名和类型，如图 3 所示。



图 3 字段设置

上传成功，可以在平台的数据源上查看数据，单击数据源操作的查看按钮如图 4 所示，数据预览如图 5 所示。



图 4 单击预览数据按钮



图 5 数据预览

### 3.1.2 新建空白工程

右击我的工程，新建一个空白的工程，如图 6 所示。

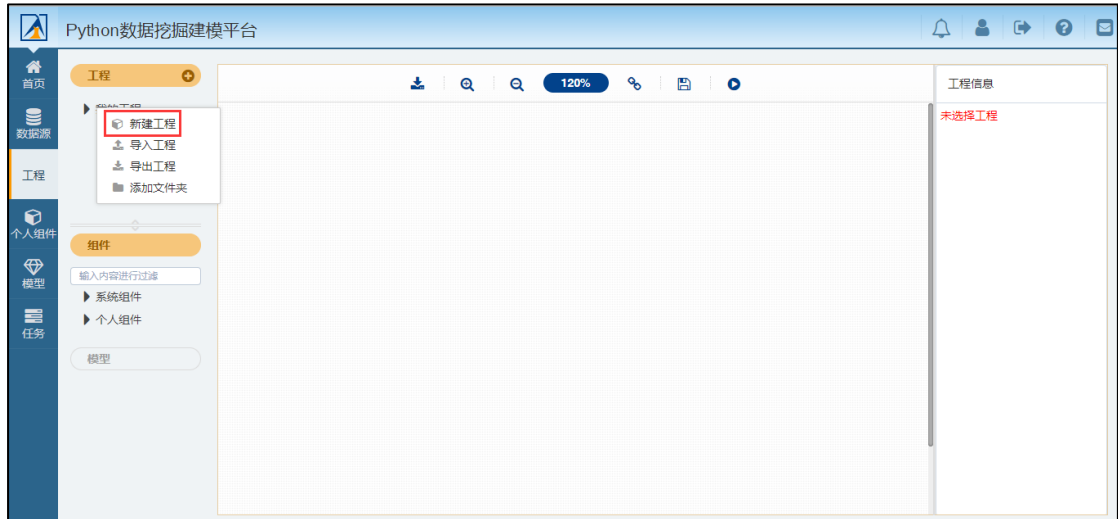


图 6 新建工程

填写工程的信息，包括工程名称和工程描述，如图 7 所示。

The image shows a 'Create Project' (创建工程) dialog box. It has a blue header with the title and a close button. The form contains three fields: 'Project Name' (工程名称) with the value 'Competitive Website User Segmentation' (竞赛网站用户分群); 'Project Description' (工程描述) with a text area containing a detailed description about user segmentation for a competitive website; and 'Project Location' (工程位置) with a dropdown menu set to 'My Projects' (我的工程). At the bottom right, there are 'Reset' (重置) and 'Confirm' (确定) buttons.

图 7 填写工程信息

## 3.2 数据预处理

读取 user\_dat 数据，步骤如图 8 所示。

- (1) 选择工程。
- (2) 选择输入源组件。
- (3) 拖入输入源组件。
- (4) 填写数据表名。

(5) 单击更新按钮，更新出数据。



图 8 输入源组件

### 3.2.1 属性规约

接下来进行属性规约。步骤如图 9 所示。

- (1) 找到预处理→数据筛选组件。
- (2) 拖入数据筛选组件，并将数据源和数据筛选组件连接。
- (3) 单击更新按钮，勾选 page\_path、username、userid、sessionid、ip、date\_time、uniquevisitorid 字段作为输出字段。
- (4) 对数据筛选组件右键，选择运行该节点。

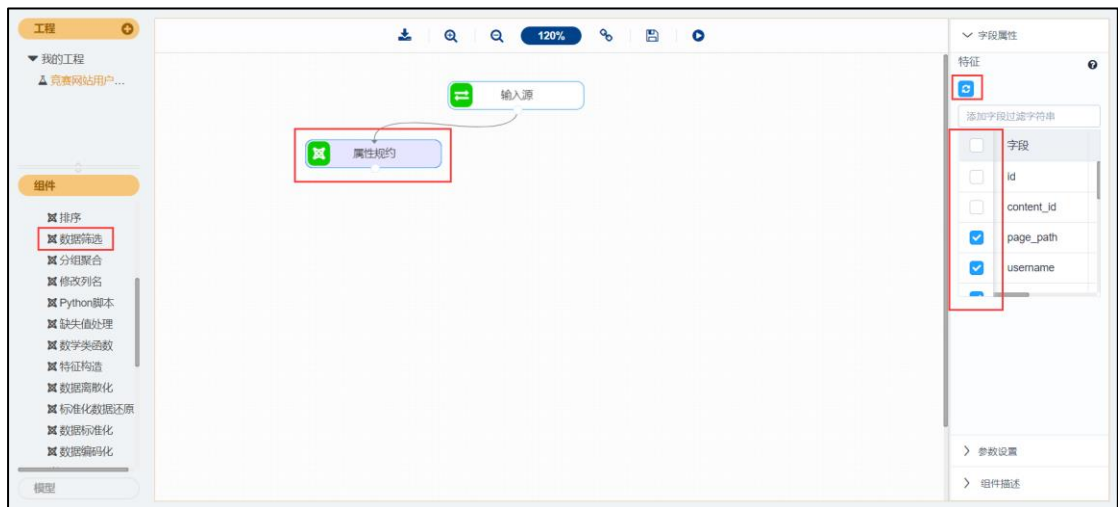


图 9 属性规约组件

(5) 运行完成后，对全表统计组件右键，选择查看数据。如图 10 所示。

page_path	username	userid	sessionid	ip	date_time	uniquevisitorid
/			D572D2142391610AED74F0BC757CDD58	60.191.123.80	2016-8-12 0:00	
/			05BB9C5E9EAB302601C4AD4F99D5AC64	64.89.234.161	2016-8-12 0:08	
/imgj/575.jhtml			EB8FBC1627FD56B64BA2E06722DBE707	66.249.65.162	2016-8-12 0:09	
/			2B9A9334275D6574C6C9E0FADEC4389B	60.191.123.80	2016-8-12 0:09	
/			7CE918EFE6BE41E05F79BE46BC80CC99	207.46.13.157	2016-8-12 0:10	
/s/jfxs/705.jhtml			0D71BB01AA9A61C07B	42.156.254.4	2016-8-12 0:15	
			578D4974B43E044279F			

图 10 属性规约结果

(6) 运行完成后，对数据筛选组件右键，重命名为属性规约。

### 3.2.2 数据变换

接下来进行数据变换，步骤如图 11 所示。

- (1) 找到预处理→Python 脚本组件。
- (2) 拖入 Python 脚本组件，并将属性规约和 Python 脚本组件连接。
- (3) 选择字段属性，在脚本处填入数据变换代码，如表 3-3 所示。
- (4) 对 Python 脚本组件右键，选择运行该节点。

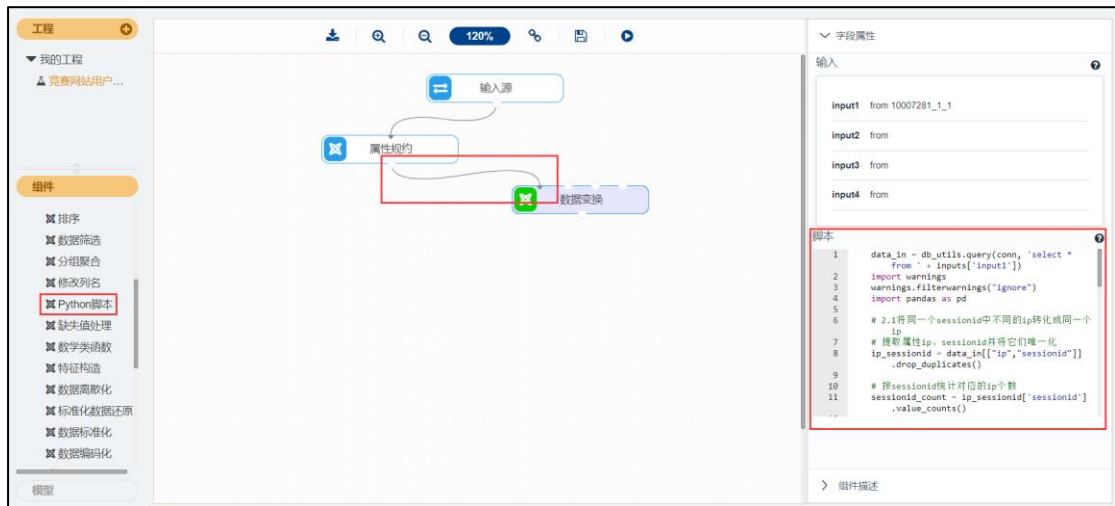


图 11 数据变换组件

表 3-3 数据变换代码

```

data_in = db_utils.query(conn, 'select * from ' + inputs['input1'])
import warnings
warnings.filterwarnings("ignore")
import pandas as pd

```

---

```
# 2.1 将同一个 sessionid 中不同的 ip 转化成同一个 ip
# 提取属性 ip、sessionid 并将它们唯一化
ip_sessionid = data_in[["ip","sessionid"]].drop_duplicates()

# 按 sessionid 统计对应的 ip 个数
sessionid_count = ip_sessionid['sessionid'].value_counts()

# 提取计数大于 1 对应的 sessionid 号
rept_sessionid = sessionid_count[sessionid_count > 1].index

# 计算满足条件的 sessionid 个数
len = rept_sessionid.size

# 同一个 sessionid 不同的 ip 用 sessionid 对应的第一个 ip 替换
for i in range(len-1):
    index = data_in[data_in['sessionid'] == rept_sessionid[i]].index
    data_in.iloc[index,]['ip'] = data_in.iloc[index[0]]['ip']

del ip_sessionid,sessionid_count,rept_sessionid,len,i

# 2.2 将一次点击中有不同的 userid 换成同一个 userid
import numpy as np

# 提取属性 userid、sessionid 并将它们唯一化
userid_sessionid = data_in[["userid","sessionid"]].drop_duplicates()

# 按 sessionid 统计对应的 ip 个数
sessionid_count_1 = userid_sessionid['sessionid'].value_counts()

# 提取计数大于 1 对应的 sessionid 号
rept_sessionid_1 = sessionid_count_1[sessionid_count_1 > 1].index

# 计算满足条件的 sessionid 个数
len_1 = rept_sessionid_1.size

# 将同一个 sessionid 且 userid 为 NA 的 userid 用该 sessionid 不为 NA 的 userid 替换
for i in range(len_1-1):
    index = data_in[data_in['sessionid'] == rept_sessionid_1[i]].index
    rept_dat = data_in.iloc[index]['userid']
    data_in.iloc[index,]['userid'] = rept_dat[rept_dat.notnull()].head(1)

del userid_sessionid,sessionid_count_1,rept_sessionid_1,len_1,i,rept_dat
```

---



```
data_out = data_in
```

```
return(data_out)
```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 12 所示。



page_path	username	userid	sessionid	ip	date_time	uniquevisitorid
/			D572D2142391610AED7 4F0BC757CDD58	60.191.123.80	2016-8-12 0:00	
/			05BB9C5E8EAB302601C 4AD4F99D5AC64	64.89.234.161	2016-8-12 0:08	
/imgj/575.jhtml			EB8FBC1627FD5B564B A2E06722DBE707	66.249.65.162	2016-8-12 0:09	
/			2B9A9334275D6574C6C 9E0FADEC4389B	60.191.123.80	2016-8-12 0:09	
/			7CE918FE6BE41E05F7 9BE46BC80CC99	207.46.13.157	2016-8-12 0:10	
/s/jfxs/705.jhtml			0D71BB01AA9A61C07B BCAE3DAD3DFFEE 578D4974B43E044279F	42.156.254.4	2016-8-12 0:15	

图 12 数据变换结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为数据变换。

### 3.2.3 用户识别

接下来进行用户识别，步骤如图 13 所示。

- (1) 找到预处理→Python 脚本组件。
- (2) 拖入 Python 脚本组件，并将数据变换和 Python 脚本组件连接。
- (3) 选择字段属性，在脚本处填入数据变换代码，如表 3-4 所示。
- (4) 对 Python 脚本组件右键，选择运行该节点。

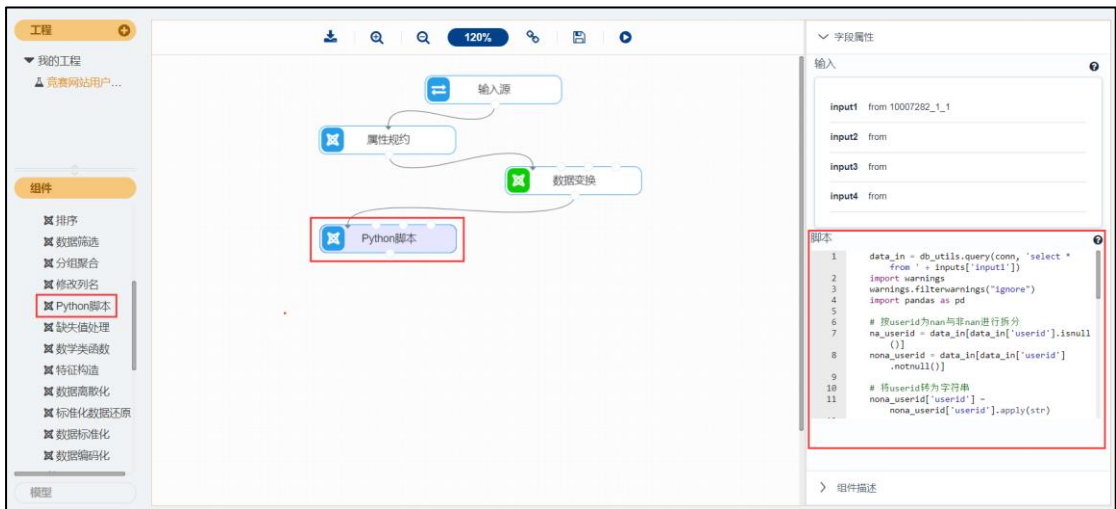


图 13 用户识别组件

表 3-4 用户识别代码

---

```
data_in = db_utils.query(conn, 'select * from ' + inputs['input1'])
import warnings
warnings.filterwarnings("ignore")
import pandas as pd

# 按 userid 为 nan 与非 nan 进行拆分
na_userid = data_in[data_in['userid'].isnull()]
nona_userid = data_in[data_in['userid'].notnull()]

# 将 userid 转为字符串
nona_userid['userid'] = nona_userid['userid'].apply(str)

# 对 nona.userid 数据集中的"userid"重新赋值
# 如果 uniqueVisitorId 不为空则用它作为 userid，若它为空则用 ip 作为 userid
# 将 na.userid 按 uniqueVisitorId 是否为空区分
na_uniqueVisitorId = na_userid[na_userid['uniquevisitorid'].isnull()]
nona_uniqueVisitorId = na_userid[na_userid['uniquevisitorid'].notnull()]

# 替换
na_uniqueVisitorId['userid'] = na_uniqueVisitorId['ip']
nona_uniqueVisitorId['userid'] = nona_uniqueVisitorId['uniquevisitorid']

# 数据整合
user_data = pd.concat([nona_userid,na_uniqueVisitorId,nona_uniqueVisitorId])

# 构造字段 reallid
userid = user_data['userid'].drop_duplicates()
userid = pd.DataFrame({'userid' : userid,
                      'reallid' : range(1,userid.size+1)})
user_data = user_data.merge(userid)

del data_in,na_userid,nona_userid,na_uniqueVisitorId,nona_uniqueVisitorId,userid

data_out = pd.DataFrame(user_data)

return(data_out)
```

---

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 14 所示。

page_path	username	userid	sessionid	ip	date_time	uniquisitorid
/index.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:24	c2a8a06c-abf5-0df d9428151f293
/zyt/index.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0df d9428151f293
/ts/747.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0df d9428151f293
/	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:26	c2a8a06c-abf5-0df d9428151f293
/w/jxq/725.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:26	c2a8a06c-abf5-0df d9428151f293
/ts/747.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6 7CF464AF4678EF8F50F	27.38.32.12	2016-8-12 1:27	c2a8a06c-abf5-0df d9428151f293 33188ba0-7880-0c

共 38334 条 25 条/页 < 1 2 3 4 5 6 ... 1534 > 前往 1 页

图 14 用户识别结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为用户识别。

### 3.2.4 行为分析

接下来进行行为分析，步骤如图 15 所示。

- (1) 找到预处理→Python 脚本组件。
- (2) 拖入 Python 脚本组件，并将用户识别和 Python 脚本组件连接。
- (3) 选择字段属性，在脚本处填入数据变换代码，如表 3-5 所示。
- (4) 对 Python 脚本组件右键，选择运行该节点。

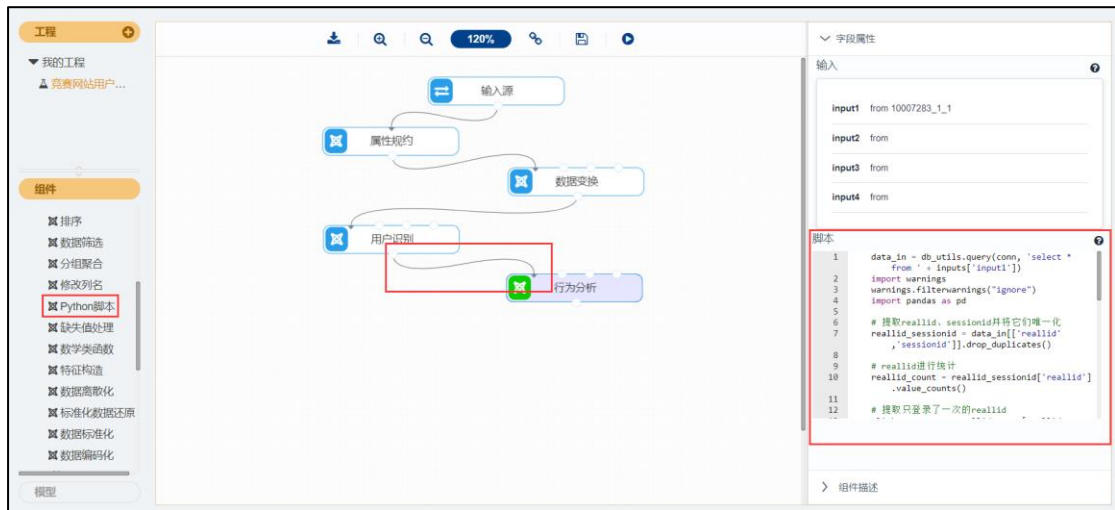


图 15 行为分析组件

表 3-5 行为分析代码

```

data_in = db_utils.query(conn, 'select * from ' + inputs['input1'])
import warnings
warnings.filterwarnings("ignore")
import pandas as pd

```

---

```
# 提取 reallid、sessionid 并将它们唯一化
reallid_sessionid = data_in[['reallid','sessionid']].drop_duplicates()

# reallid 进行统计
reallid_count = reallid_sessionid['reallid'].value_counts()

# 提取只登录了一次的 reallid
click_con_user = reallid_count[reallid_count == 1].index

# 提取登录一次用户的原始点击数据
click_con_data = data_in[data_in['reallid']].apply(lambda x:x in click_con_user)

# 按 click.one.data 中的 reallID 进行统计
reallid_count_1 = click_con_data['reallid'].value_counts()

# 提取只登录了一次且只点击了一个网页的用户
one_click_user = reallid_count_1[reallid_count_1 == 1].index

# 从原始数据中去掉闲逛人员的数据
# 提取用户编号并唯一化
user = data_in['reallid'].drop_duplicates()

# 提取非"闲逛人员"的用户编号
user_1 = pd.DataFrame(list(set(user).difference(one_click_user)))
user_1.columns = ['reallid']

# 提取非"闲逛人员"的原始数据
new_data = pd.merge(data_in,user_1,how='inner',on='reallid')

del data_in,reallid_sessionid,reallid_count,click_con_user,\
    click_con_data,reallid_count_1,one_click_user,user,user_1

data_out = pd.DataFrame(new_data)

return(data_out)
```

---

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 16 所示。

page_path	username	userid	sessionid	ip	date_time	uniquevisitorid
/index.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:24	c2a8a06c-abf5-0df d9428151f293
/zyfj/index.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0df d9428151f293
/ts/747.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0df d9428151f293
/	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:26	c2a8a06c-abf5-0df d9428151f293
/wjqx/725.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:26	c2a8a06c-abf5-0df d9428151f293
/ts/747.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:27	c2a8a06c-abf5-0df d9428151f293
			7CF464AF4678EF8F50F			33188ba0-7880-0c

共 34739 条 | 25 条页 | < 1 2 3 4 5 6 ... 1390 > 前往 1 页

图 16 行为分析结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为行为分析。

### 3.2.5 网址分析

接下来进行网址分析，步骤如图 17 所示。

- (1) 找到预处理→Python 脚本组件。
- (2) 拖入 Python 脚本组件，并将行为分析和 Python 脚本组件连接。
- (3) 选择字段属性，在脚本处填入数据变换代码，如表 3-6 所示。
- (4) 对 Python 脚本组件右键，选择运行该节点。

The screenshot shows a workflow editor with several components: '输入源' (Input Source), '属性规约' (Attribute Reduction), '数据变换' (Data Transformation), '用户识别' (User Identification), '行为分析' (Behavior Analysis), and '网址分析' (URL Analysis). The 'Python 脚本' (Python Script) component is highlighted in the left sidebar. The right panel shows the script code for the '行为分析' component.

```

脚本
1 data_in = db_utils.query(conn, 'select *
2   from ' + inputs['input1'])
3 import warnings
4 warnings.filterwarnings("ignore")
5 import pandas as pd
6 # 新增一类序号
7 data_in['number_id'] = range(1, len(data_in)
8   )+1
9 # 提取 .html 结尾的数据
10 html_data = data_in[data_in['page_path']
11   .apply(lambda x:x.endswith('.html'))]
12 # 提取非 .html 结尾的数据

```

图 17 网址分析组件

表 3-6 网址分析代码

```

data_in = db_utils.query(conn, 'select * from ' + inputs['input1'])
import warnings
warnings.filterwarnings("ignore")
import pandas as pd

```

```

# 新增一类序号
data_in['number_id'] = range(1,len(data_in)+1)

# 提取.jhtml 结尾的数据
jhtml_data = data_in[data_in['page_path'].apply(lambda x:x.endswith('.jhtml'))]

# 提取非.jhtml 结尾的数据
#unjhtml_data = data_in[-data_in['number_id'].isin(jhtml_data['number_id'])]

# 统计各非 jhtml 网址的个数
#unjhtml_count = unjhtml_data['page_path'].value_counts()

data_out = pd.DataFrame(jhtml_data)

return(data_out)

```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 18 所示。

page_path	username	userid	sessionid	ip	date_time	uniquevisitorid
/index.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:24	c2a8a06c-abf5-0d6 d9428151f293
/zytj/index.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0d6 d9428151f293
/ts/747.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0d6 d9428151f293
/wjxq/725.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:26	c2a8a06c-abf5-0d6 d9428151f293
/ts/747.jhtml	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:27	c2a8a06c-abf5-0d6 d9428151f293
/zytj/index.jhtml	gangzhi21cn	9384	7CF464AF4678EF8F50F 2AC7FBD9C93852 7CF464AF4678EF8F50F	10.130.142.131	2016-8-12 8:48	33188ba0-7880-0c 3-4dc196d459f6 33188ba0-7880-0c

图 18 网址分析结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为网址分析。

### 3.2.6 数据清洗

接下来进行数据清洗，步骤如图 19 所示。

- (1) 找到预处理→Python 脚本组件。
- (2) 拖入 Python 脚本组件，并将网址分析和 Python 组件连接。
- (3) 选择字段属性，在脚本处填入数据变换代码，如表 3-7 所示。
- (4) 对 Python 脚本组件右键，选择运行该节点。

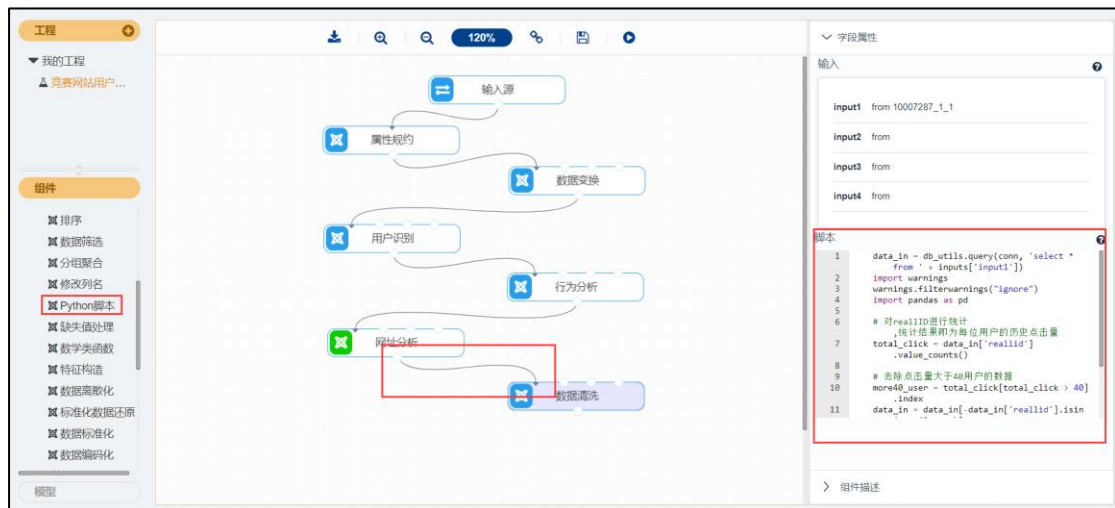


图 19 数据清洗组件

表 3-7 数据清洗代码

```

data_in = db_utils.query(conn, 'select * from ' + inputs['input1'])
import warnings
warnings.filterwarnings("ignore")
import pandas as pd

# 对 realIID 进行统计,统计结果即为每位用户的历史点击量
total_click = data_in['realIID'].value_counts()

# 去除点击量大于 40 用户的数据
more40_user = total_click[total_click > 40].index
data_in = data_in[-data_in['realIID'].isin(more40_user)]

del total_click, more40_user

data_out = pd.DataFrame(data_in)

return(data_out)

```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 20 所示。

page_path	username	userid	sessionid	ip	date_time	uniquevisitorid
/index.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:24	c2a8a06c-abf5-0df d9428151f293
/zytj/index.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0df d9428151f293
/ts/747.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:25	c2a8a06c-abf5-0df d9428151f293
/vjqxq/725.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:26	c2a8a06c-abf5-0df d9428151f293
/ts/747.html	yyang891010	9383	8F63C7A49D56BDA9706 68D57B43D1DF6	27.38.32.12	2016-8-12 1:27	c2a8a06c-abf5-0df d9428151f293
/zytj/index.html	gangzhi21cn	9384	7CF464AF4678EF8F50F 2AC7FBDC93852 7CF464AF4678EF8F50F	10.130.142.131	2016-8-12 8:48	33188ba0-7880-0c 3-4dc196d459f6 33188ba0-7880-0c

共 16673 条 25 条/页 < 1 2 3 4 5 6 ... 667 > 前往 1 页

图 20 数据清洗结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为数据清洗。

### 3.2.7 网页分析

接下来进行数据清洗，步骤如图 21 所示。

- (1) 找到预处理→Python 脚本组件。
- (2) 拖入 Python 脚本组件，并将数据清洗和 Python 组件连接。
- (3) 选择字段属性，在脚本处填入数据变换代码，如表 3-8 所示。
- (4) 对 Python 脚本组件右键，选择运行该节点。

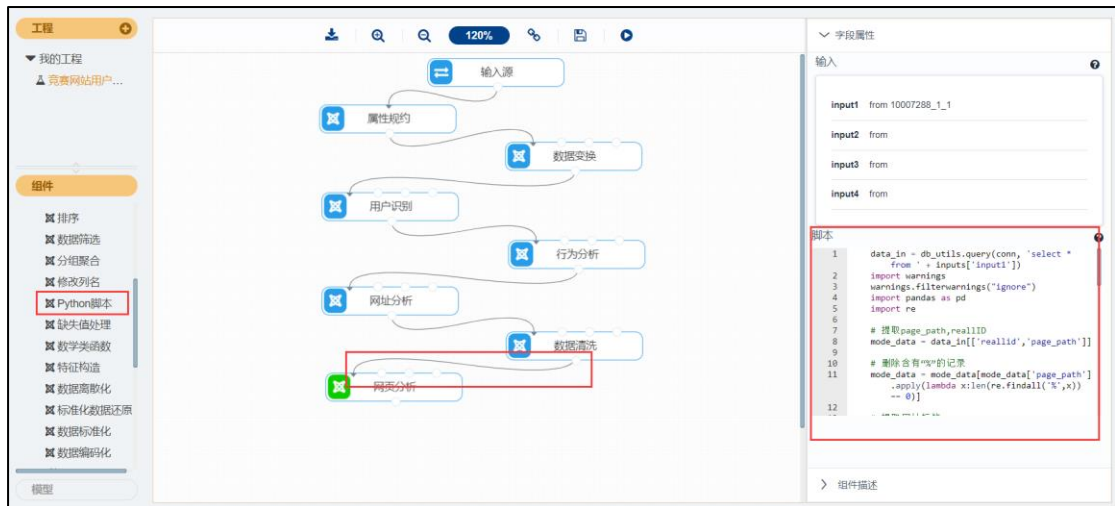


图 21 网页分析组件

表 3-8 网页分析代码

```
data_in = db_utils.query(conn, 'select * from ' + inputs['input1'])
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import re

# 提取 page_path,reallID
mode_data = data_in[['reallid','page_path']]

# 删除含有“%”的记录
mode_data = mode_data[mode_data['page_path'].apply(lambda x:len(re.findall('%',x)) == 0)]

# 提取网址标签
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('.jhtml',''))
```



```

mode_data['page_path'] = mode_data['page_path'].apply(lambda x:re.sub('\d',"x))
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('index',''))
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('/','))
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('_','))

# 删除空数据 主页/index
mode_data = mode_data[mode_data['page_path'] != ""]

# 统计网页内容符号数，删除一些异常的网页内容符号如：sdfasf、cook...
url_conent = mode_data['page_path'].value_counts()
url = url_conent.index

error_url = ['dsjfkf','sjfxs','jingsa','sdfasf','cookie','jiao','jao','asdf','sjsdf','jmg']
url = url[~url.isin(error_url)]

## 网页汇总
mode_data = mode_data[mode_data['page_path'].isin(url)]

mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('firsttipdm','yxzp'))
mode_data['page_path'] = mode_data['page_path'].apply(lambda
x:x.replace('secondtipdm','yxzp'))
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('thirdtipdm','yxzp'))
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('fourthtipdm','yxzp'))
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('sm','jszz'))
mode_data['page_path'] = mode_data['page_path'].apply(lambda x:x.replace('td','jszz'))

mode_data = mode_data.groupby(['reallid','page_path']).size().unstack().fillna(0).reset_index()

del url_conent,url,error_url

data_out = pd.DataFrame(mode_data)

return(data_out)

```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 22 所示。

reallid	cgal	information	jmgj	jszz	jxsp	kxkm
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	1	0	0	5	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0

图 22 网页分析结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为网页分析。

### 3.2.8 属性构造

接下来进行数据清洗，步骤如图 23 所示。

- (1) 找到预处理→Python 脚本组件。
- (2) 拖入 Python 脚本组件，并将网页分析和 Python 组件连接。
- (3) 选择字段属性，在脚本处填入数据变换代码，如表 3-9 所示。
- (4) 对 Python 脚本组件右键，选择运行该节点。

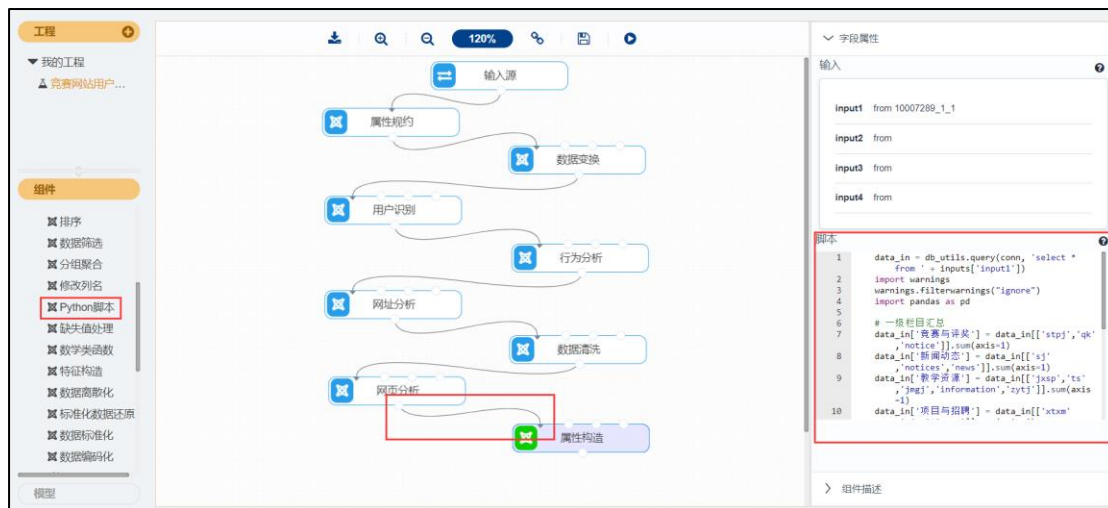


图 23 属性构造组件

表 3-9 属性构造代码

```
data_in = db_utils.query(conn, 'select * from ' + inputs['input1'])
import warnings
warnings.filterwarnings("ignore")
import pandas as pd

# 一级栏目汇总
data_in['竞赛与评奖'] = data_in[['stpj','qk','notice']].sum(axis=1)
data_in['新闻动态'] = data_in[['sj','notices','news']].sum(axis=1)
data_in['教学资源'] = data_in[['xsp','ts','jmgj','information','zytj']].sum(axis=1)
data_in['项目与招聘'] = data_in[['xtxm','wjxq','zxns']].sum(axis=1)
data_in['创新与合作'] = data_in[['cgal','kjxm','qyal','zzszl']].sum(axis=1)
data_in['优秀作品'] = data_in[['yxzp']].sum(axis=1)
data_in['竞赛组织'] = data_in[['jszz']].sum(axis=1)
```

```

# 提取建模数据，即：建模样本表
data_in = data_in[['reallid','竞赛与评奖','新闻动态','教学资源','项目与招聘','创新与合作','优秀作品','竞赛组织']]
data_in.rename(columns={'reallid':'用户编号'},inplace = True)

data_out = pd.DataFrame(data_in)

return(data_out)

```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 24 所示。

用户编号	竞赛与评奖	新闻动态	教学资源	项目与招聘	创新与合作	优秀作品
1	0	0	3	1	0	0
2	0	0	3	0	0	0
3	0	0	2	0	0	2
4	0	0	2	0	0	0
5	9	18	1	1	1	0
6	0	0	1	0	0	0
7	0	0	1	0	0	0
8	0	0	1	0	0	0

图 24 属性构造结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为属性构造。

## 3.3 模型构建

### 3.3.1 K-Means 聚类算法

选择 K-Means 聚类算法模型，步骤如如图 25、图 26 所示。

- (1) 找到聚类→K-Means 组件。
- (2) 拖入 K-Means 组件，将属性构造和 K-Means 组件连接。
- (3) 选择字段属性，单击更新数据，勾选除了用户编号外的全部字段。
- (4) 选择参数设置，设置聚类数（n\_clusters）的值为 5，其他的参数都设置为默认值。

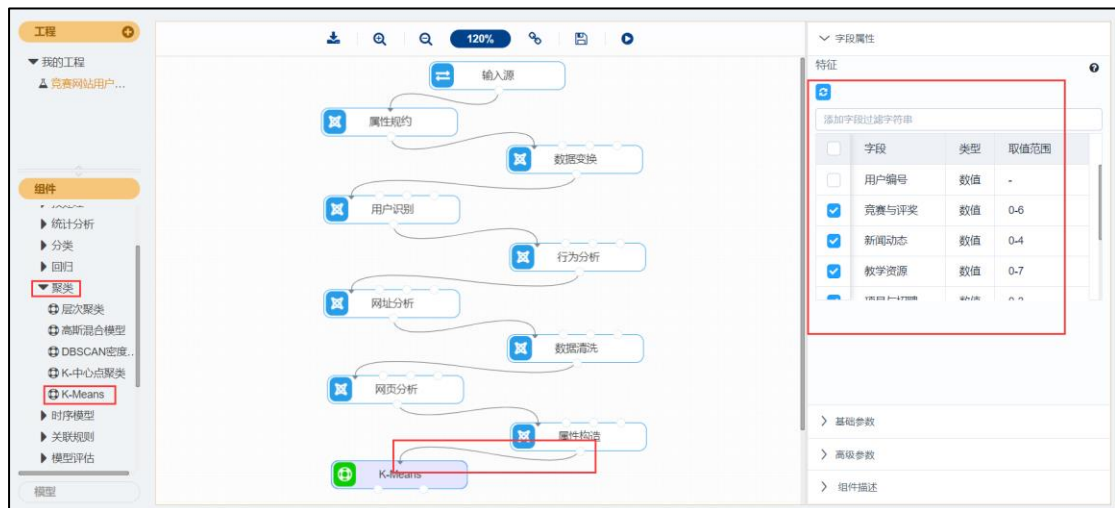


图 25 M-Means 聚类组件\_字段属性

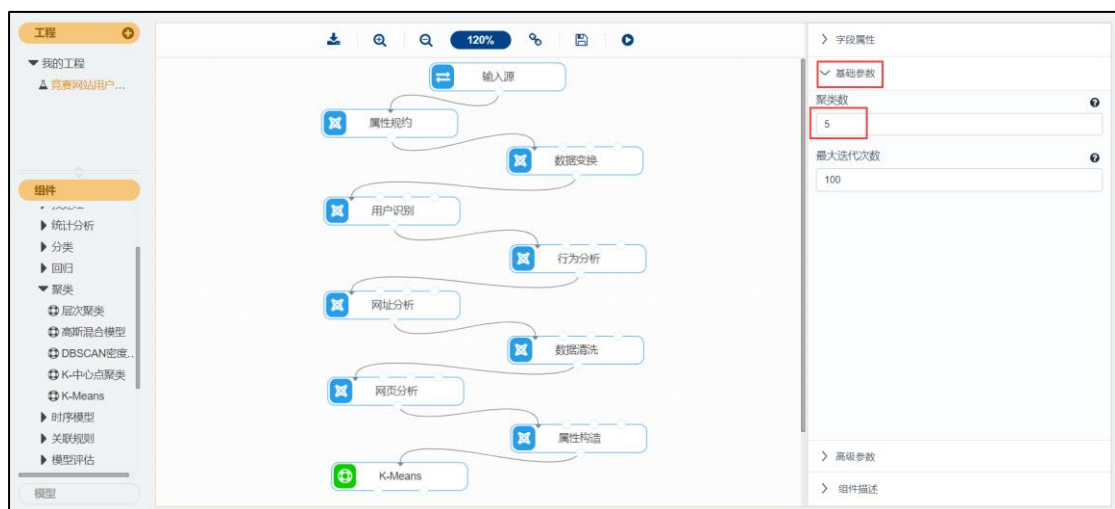


图 26 K-Means 组件\_参数设置

(5) 运行完成后，对 K-Means 组件右键，选择查看数据，K-Means 的输出表结果如图 27 所示。选择查看报告，K-Means 的报告如图 28 所示。

新闻动态	教学资源	项目与招聘	创新与合作	优秀作品	竞赛组织	cluster_id
0	3	1	0	0	0	1
0	3	0	0	0	0	1
0	2	0	0	2	0	2
0	2	0	0	0	0	2
18	1	1	1	0	5	4
0	1	0	0	0	0	2
0	1	0	0	0	0	2
0	1	0	0	0	0	2

共 2826 条 25 条/页 < 1 2 3 4 5 6 ... 114 > 前往 1 页

图 27 K-Means 聚类算法的结果



图 28 K-Means 聚类算法的报告