目录

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams["font.sans-serif"] = "SimHei" #解决中文乱码问题
import seaborn as sns
import random
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn import model_selection
from sklearn.neighbors import KNeighborsRegressor
```

缺失值查看预处理

```
df_train =
pd.read_csv(r'C:\Users\gaohao\Downloads\data_format1\train_format1.csv')
df_test = pd.read_csv(r'C:\Users\gaohao\Downloads\data_format1\test_format1.csv')
user_info =
pd.read_csv(r'C:\Users\gaohao\Downloads\data_format1\user_info_format1.csv')
user_log =
pd.read_csv(r'C:\Users\gaohao\Downloads\data_format1\user_log_format1.csv')
print(df_test.shape,df_train.shape)
print(user_info.shape,user_log.shape)
(261477, 3) (260864, 3)

(424170, 3) (54925330, 7)
user_info.info()
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 424170 entries, 0 to 424169

Data columns (total 3 columns):

user_id      424170 non-null int64

age_range    421953 non-null float64

gender       417734 non-null float64

dtypes: float64(2), int64(1)

memory usage: 9.7 MB
user_info.head(10)
```

[6]:

,,

	user_id	age_range	gender
0	376517	6.0	1.0
1	234512	5.0	0.0
2	344532	5.0	0.0
3	186135	5.0	0.0
4	30230	5.0	0.0
5	272389	6.0	1.0
6	281071	4.0	0.0
7	139859	7.0	0.0
8	198411	5.0	1.0
9	67037	4.0	1.0

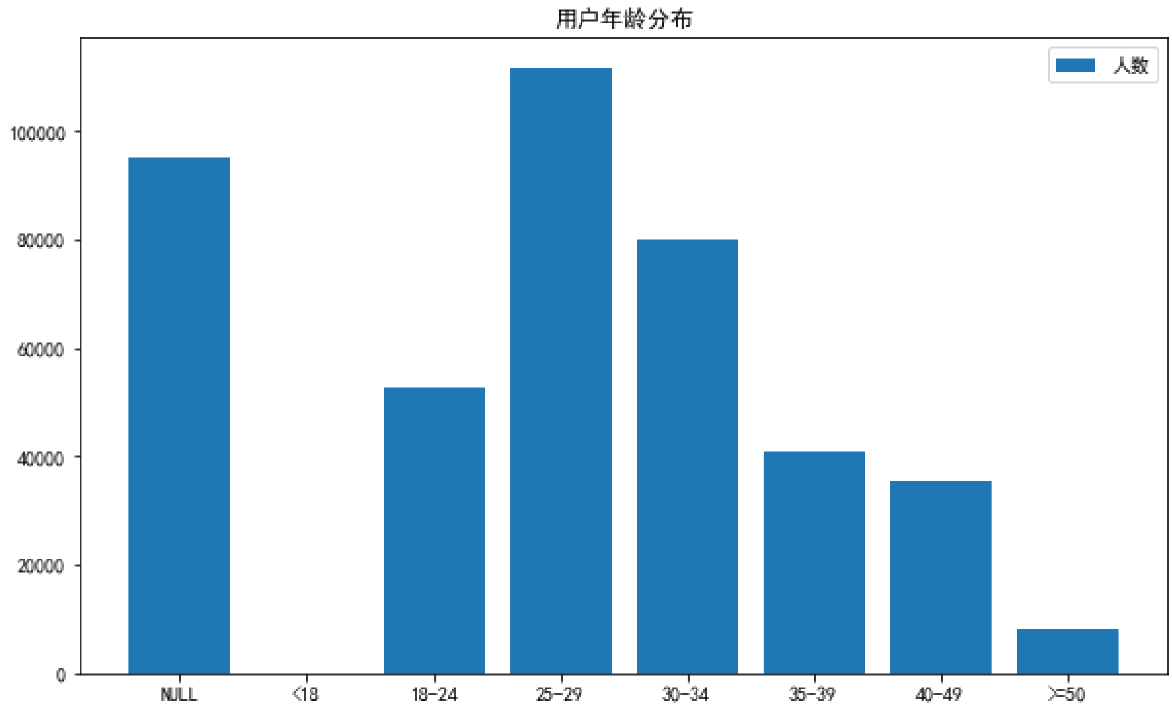
,

```
user_info['age_range'].replace(0.0,np.nan,inplace=True)
user_info['gender'].replace(2.0,np.nan,inplace=True)
user_info.info()
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 424170 entries, 0 to 424169

Data columns (total 3 columns):

user_id      424170 non-null int64

age_range    329039 non-null float64

gender       407308 non-null float64

dtypes: float64(2), int64(1)

memory usage: 9.7 MB
user_info['age_range'].replace(np.nan,-1,inplace=True)
user_info['gender'].replace(np.nan,-1,inplace=True)
fig = plt.figure(figsize = (10, 6))
x = np.array(["NULL","<18","18-24","25-29","30-34","35-39","40-49",">=50"])
#<18岁为1；[18,24]为2； [25,29]为3； [30,34]为4；[35,39]为5；[40,49]为6； > = 50时为7
和8
y = np.array([user_info[user_info['age_range'] == -1]['age_range'].count(),
             user_info[user_info['age_range'] == 1]['age_range'].count(),
             user_info[user_info['age_range'] == 2]['age_range'].count(),
             user_info[user_info['age_range'] == 3]['age_range'].count(),
             user_info[user_info['age_range'] == 4]['age_range'].count(),
             user_info[user_info['age_range'] == 5]['age_range'].count(),
             user_info[user_info['age_range'] == 6]['age_range'].count(),
```

```
              user_info[user_info['age_range'] == 7]['age_range'].count() +
user_info[user_info['age_range'] == 8]['age_range'].count()])
plt.bar(x,y,label='人数')
plt.legend()
plt.title('用户年龄分布')
```

[9]:

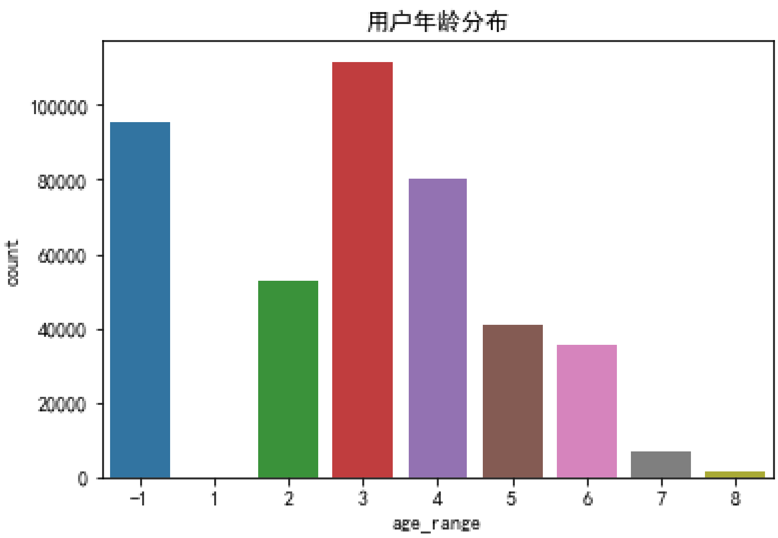
```
Text(0.5, 1.0, '用户年龄分布')
```




```
sns.countplot(x = 'age_range', order = [-1,1,2,3,4,5,6,7,8], data = user_info)
plt.title('用户年龄分布')
```

[10]:

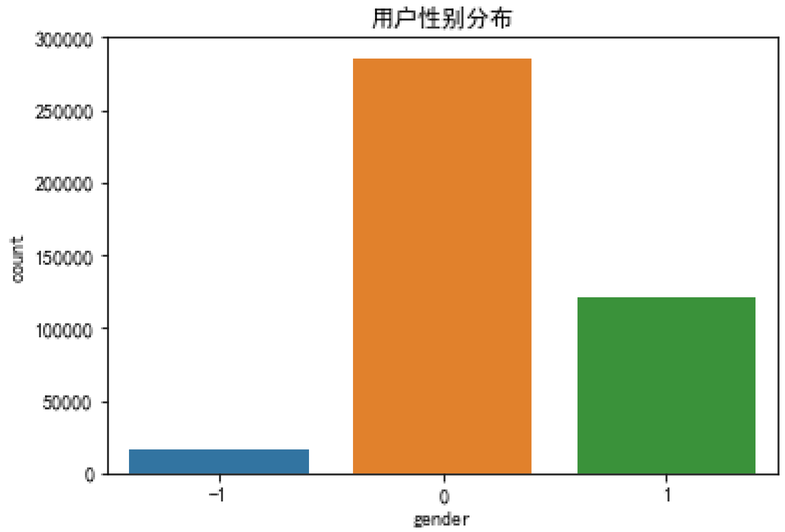
```
Text(0.5, 1.0, '用户年龄分布')
```



```
sns.countplot(x='gender',order = [-1,0,1],data = user_info)
plt.title('用户性别分布')
```

[11]:

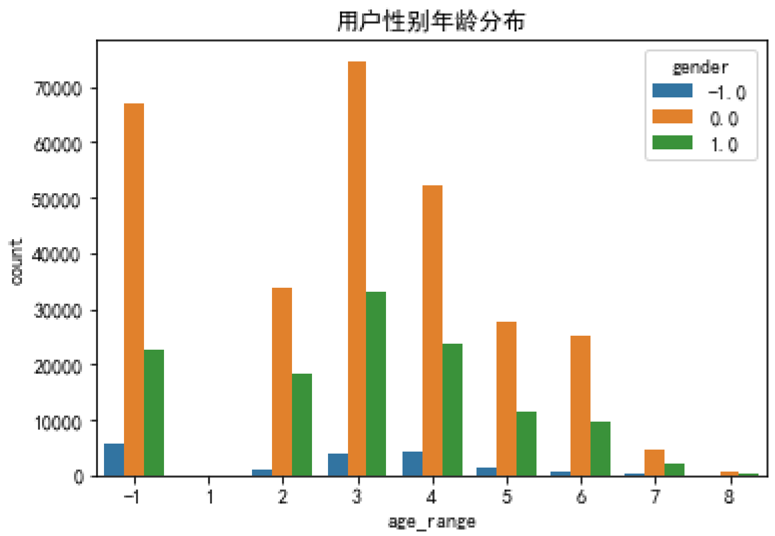
```
Text(0.5, 1.0, '用户性别分布')
```




```
sns.countplot(x = 'age_range', order = [-1,1,2,3,4,5,6,7,8],hue= 'gender',data =
user_info)
plt.title('用户性别年龄分布')
```

[12]:

```
Text(0.5, 1.0, '用户性别年龄分布')
```

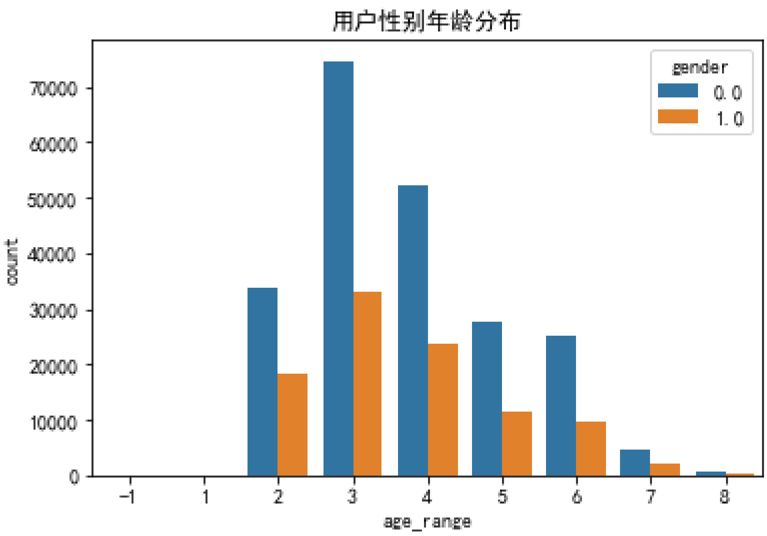


- 年纪的缺省值不少，性别的缺省值倒是不多。
- 用户年纪主要分布在18-34岁，且主要为女性。
- 缺失值处理的话，先简单处理一下，把缺失值都做删除处理吧，后面继续尝试的话可以试试填充缺失值
- 后来又注释掉了，没有删，因为这里是原始数据，应该在建立好特征之后再删吧

```
user_info['age_range'].replace(-1,np.nan,inplace=True)
user_info['gender'].replace(-1,np.nan,inplace=True)
#user_info = user_info.dropna()
#user_info.info()
sns.countplot(x = 'age_range', order = [-1,1,2,3,4,5,6,7,8],hue= 'gender',data =
user_info)
plt.title('用户性别年龄分布')
```

[15]:

```
Text(0.5, 1.0, '用户性别年龄分布')
```




```
user_log.head()
```

[16]:

,,,

	user_id	item_id	cat_id	seller_id	brand_id	time_stamp	action_type
0	328862	323294	833	2882	2661.0	829	0
1	328862	844400	1271	2882	2661.0	829	0
2	328862	575153	1271	2882	2661.0	829	0
3	328862	996875	1271	2882	2661.0	829	0
4	328862	1086186	1271	1253	1049.0	829	0

,

```
user_log.isnull().sum(axis=0)
```

[17]:

```
user_id           0
,item_id           0
,cat_id            0
,seller_id         0
,brand_id      91015
,time_stamp        0
,action_type       0
,dtype: int64
#user_log = user_log.dropna()
user_log.isnull().sum(axis=0)
```

[18]:

```
user_id           0
,item_id           0
,cat_id            0
,seller_id         0
,brand_id      91015
,time_stamp        0
,action_type       0
,dtype: int64
user_log.info()
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 54925330 entries, 0 to 54925329

Data columns (total 7 columns):

user_id        int64

item_id        int64

cat_id         int64

seller_id      int64

brand_id       float64

time_stamp     int64

action_type    int64

dtypes: float64(1), int64(6)

memory usage: 2.9 GB
```

- 这里对于用户日志里的商品品牌的缺失也做了删除处理，反正也不多是不是
- 没删，没删

初步可视化

- user_log前面几行全是编码，购物者的唯一ID编码，商品的唯一编码，商品所属品类的唯一编码，商家的唯一ID编码，商品品牌的唯一编码

- 后面是购买时间，与活动日志记录

```
df_train.head(10)
```

[20]:

,,

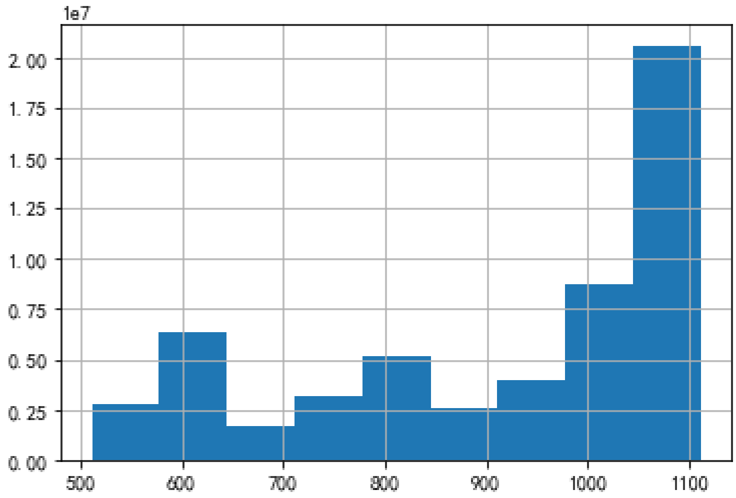
	user_id	merchant_id	label
0	34176	3906	0
1	34176	121	0
2	34176	4356	1
3	34176	2217	0
4	230784	4818	0
5	362112	2618	0
6	34944	2051	0
7	231552	3828	1
8	231552	2124	0
9	232320	1168	0

,

```
df_train.info()
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 260864 entries, 0 to 260863

Data columns (total 3 columns):

user_id        260864 non-null int64

merchant_id    260864 non-null int64

label          260864 non-null int64

dtypes: int64(3)

memory usage: 6.0 MB
user_log['time_stamp'].hist(bins = 9)
```

[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a59a849438>
```

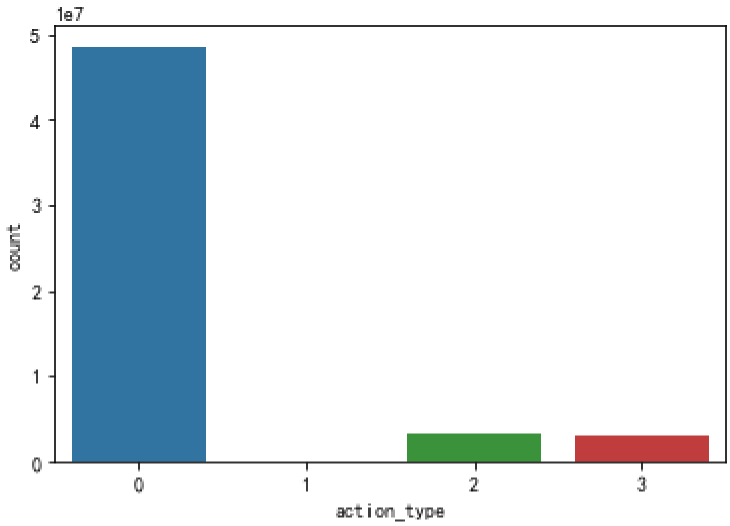


- 618和双十一购买的东西最多

```
sns.countplot(x = 'action_type', order = [0,1,2,3],data = user_log)
```

[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a59a9b9c50>
```



- 绝大多数都是单击，加入购物车的动作很少，比购买和收藏的动作还要少

特征工程

```
df_train[df_train['label'] == 1]
```

[24]:

,,

,,

,,

,,

,,,,,,,,,,,,,,,

	user_id	merchant_id	label
2	34176	4356	1
7	231552	3828	1
53	306816	1489	1
57	176256	3323	1
59	307584	1340	1
69	309120	4775	1
89	247680	3990	1
90	182400	3938	1
102	53376	839	1
150	127104	4048	1
153	324480	2891	1
154	259200	4048	1
195	269184	4992	1
196	7296	4976	1
204	205440	3472	1
224	340608	2537	1
239	408192	1093	1
305	28032	4044	1
321	97152	361	1
333	165249	1487	1
346	363393	1255	1
361	104577	4005	1
367	368001	2217	1
390	43905	3835	1
392	44673	4976	1
401	111489	608	1
417	114561	4043	1
432	117633	2403	1
439	250497	3889	1
440	250497	3319	1

	user_id	merchant_id	label
...	...	...	...
260298	319358	191	1
260306	59774	1393	1
260308	387710	10	1
260312	388478	415	1
260337	261758	4536	1
260346	638	191	1
260366	136574	641	1
260405	274814	3363	1
260413	341630	2592	1
260422	212606	1892	1
260427	409982	4525	1
260449	87422	503	1
260453	88190	3266	1
260485	28286	1203	1
260504	162686	1885	1
260515	165503	4701	1
260593	376703	1031	1
260597	311423	3936	1
260603	312959	2928	1
260613	52607	4648	1
260615	53375	3828	1
260618	316031	2031	1
260624	120959	3323	1
260694	133247	1346	1
260720	268415	2397	1
260747	208511	2592	1
260793	87935	1964	1
260794	87935	3734	1
260799	350591	4394	1

	user_id	merchant_id	label
260842	422783	2026	1

,

15952 rows × 3 columns

,

```
user_log[(user_log['user_id'] == 34176) & (user_log['seller_id'] == 3906)]
```

[25]:

,,,
,,,
,,,
,,,
,,,

	user_id	item_id	cat_id	seller_id	brand_id	time_stamp	action_type
35905644	34176	757713	821	3906	6268.0	1110	0
35905646	34176	757713	821	3906	6268.0	1110	0
35905672	34176	757713	821	3906	6268.0	1110	0
35905696	34176	718096	1142	3906	6268.0	1031	3
35905720	34176	757713	821	3906	6268.0	1031	3
35905791	34176	613698	821	3906	6268.0	1021	0
35905804	34176	757713	821	3906	6268.0	1108	0
35905824	34176	757713	821	3906	6268.0	1029	0
35905830	34176	1093165	1397	3906	6268.0	1027	0
35905831	34176	898580	662	3906	6268.0	1027	0
35905832	34176	569051	1577	3906	6268.0	1027	0
35905834	34176	185202	821	3906	6268.0	1027	0
35905876	34176	757713	821	3906	6268.0	1111	2
35905886	34176	757713	821	3906	6268.0	1111	0
35905895	34176	757713	821	3906	6268.0	1111	0
35905900	34176	757713	821	3906	6268.0	1111	0
35905960	34176	757713	821	3906	6268.0	1107	0
35905966	34176	757713	821	3906	6268.0	1107	0
35905995	34176	965699	662	3906	6268.0	1027	0
35905996	34176	187402	1577	3906	6268.0	1027	0
35905999	34176	757713	821	3906	6268.0	1027	0

	user_id	item_id	cat_id	seller_id	brand_id	time_stamp	action_type
35906001	34176	569051	1577	3906	6268.0	1027	0
35906005	34176	702940	662	3906	6268.0	1027	0
35906007	34176	832131	662	3906	6268.0	1027	0
35906008	34176	48054	302	3906	6268.0	1027	0
35906009	34176	757713	821	3906	6268.0	1027	0
35906011	34176	320263	662	3906	6268.0	1027	0
35906012	34176	468438	821	3906	6268.0	1027	0
35906013	34176	963534	821	3906	6268.0	1027	0
35906014	34176	613698	821	3906	6268.0	1027	0
35906016	34176	757713	821	3906	6268.0	1027	0
35906017	34176	963534	821	3906	6268.0	1027	0
35906020	34176	157439	662	3906	6268.0	1027	0
35906021	34176	475546	1397	3906	6268.0	1027	0
35906023	34176	246109	821	3906	6268.0	1027	0
35906025	34176	757713	821	3906	6268.0	1027	0
35906026	34176	523545	662	3906	6268.0	1027	0
35906027	34176	198962	1577	3906	6268.0	1027	0
35906085	34176	613698	821	3906	6268.0	1020	0

,

想要建立的特征

需要根据user_id，和merchant_id（seller_id）,从用户画像表以及用户日志表中提取特征，填写到df_train这个数据框中，从而训练评估模型 需要建立的特征如下：

- 用户的年龄(age_range)
- 用户的性别(gender)
- 某用户在该商家日志的总条数(total_logs)
- 用户浏览的商品的数目，就是浏览了多少个商品(unique_item_ids)
- 浏览的商品的种类的数目，就是浏览了多少种商品(categories)
- 用户浏览的天数(browse_days)
- 用户单击的次数(one_clicks)
- 用户添加购物车的次数(shopping_carts)
- 用户购买的次数(purchase_times)
- 用户收藏的次数(favourite_times)

```
df_train.head()
```

[26]:

, ,

	user_id	merchant_id	label
0	34176	3906	0
1	34176	121	0
2	34176	4356	1
3	34176	2217	0
4	230784	4818	0

,

```
user_info.head()
```

[27]:

, ,

	user_id	age_range	gender
0	376517	6.0	1.0
1	234512	5.0	0.0
2	344532	5.0	0.0
3	186135	5.0	0.0
4	30230	5.0	0.0

,

```
user_log.head()
```

[28]:

, ,

	user_id	item_id	cat_id	seller_id	brand_id	time_stamp	action_type
0	328862	323294	833	2882	2661.0	829	0
1	328862	844400	1271	2882	2661.0	829	0
2	328862	575153	1271	2882	2661.0	829	0
3	328862	996875	1271	2882	2661.0	829	0
4	328862	1086186	1271	1253	1049.0	829	0

,

age_range,gender特征添加

```
df_train = pd.merge(df_train,user_info,on="user_id",how="left")
df_train.head()
```

[29]:

,,,

	user_id	merchant_id	label	age_range	gender
0	34176	3906	0	6.0	0.0
1	34176	121	0	6.0	0.0
2	34176	4356	1	6.0	0.0
3	34176	2217	0	6.0	0.0
4	230784	4818	0	NaN	0.0

,

total_logs特征添加

```
total_logs_temp =
user_log.groupby([user_log["user_id"],user_log["seller_id"]]).count().reset_index
()[["user_id","seller_id","item_id"]]
total_logs_temp.head(10)
```

[31]:

,,

	user_id	seller_id	item_id
0	1	471	1
1	1	739	1
2	1	925	4
3	1	1019	14
4	1	1156	1
5	1	2245	5
6	1	4026	5
7	1	4177	1
8	1	4335	1
9	2	420	26

,

```
total_logs_temp.rename(columns=
{"seller_id":"merchant_id","item_id":"total_logs"},inplace=True)
total_logs_temp.head()
```

[32]:

,,

	user_id	merchant_id	total_logs
0	1	471	1
1	1	739	1
2	1	925	4
3	1	1019	14
4	1	1156	1

,

```
df_train = pd.merge(df_train,total_logs_temp,on=
["user_id","merchant_id"],how="left")
df_train.head()
```

[33]:

,,

	user_id	merchant_id	label	age_range	gender	total_logs
0	34176	3906	0	6.0	0.0	39
1	34176	121	0	6.0	0.0	14
2	34176	4356	1	6.0	0.0	18
3	34176	2217	0	6.0	0.0	2
4	230784	4818	0	NaN	0.0	8

,

unique_item_ids特征添加

```
unique_item_ids_temp =
user_log.groupby([user_log["user_id"],user_log["seller_id"],user_log["item_id"]])
.count().reset_index()[["user_id","seller_id","item_id"]]
unique_item_ids_temp.head(10)
```

[35]:

,,

	user_id	seller_id	item_id
0	1	471	638653
1	1	739	556107
2	1	925	504149
3	1	1019	1110495
4	1	1156	896183
5	1	2245	181459
6	1	2245	452837
7	1	2245	543397
8	1	2245	779078
9	1	4026	112203

,

```
unique_item_ids_temp1 =
unique_item_ids_temp.groupby([unique_item_ids_temp["user_id"],unique_item_ids_tem
p["seller_id"]]).count().reset_index()
unique_item_ids_temp1.head(10)
```

[37]:

,,

	user_id	seller_id	item_id
0	1	471	1
1	1	739	1
2	1	925	1
3	1	1019	1
4	1	1156	1
5	1	2245	4
6	1	4026	1
7	1	4177	1
8	1	4335	1
9	2	420	15

,

```
unique_item_ids_temp1.rename(columns=
{"seller_id":"merchant_id","item_id":"unique_item_ids"},inplace=True)
unique_item_ids_temp1.head(10)
```

[39]:

, ,

	user_id	merchant_id	unique_item_ids
0	1	471	1
1	1	739	1
2	1	925	1
3	1	1019	1
4	1	1156	1
5	1	2245	4
6	1	4026	1
7	1	4177	1
8	1	4335	1
9	2	420	15

,

```
df_train = pd.merge(df_train,unique_item_ids_temp1,on=
["user_id","merchant_id"],how="left")
df_train.head()
```

[40]:

, ,

	user_id	merchant_id	label	age_range	gender	total_logs	unique_item_ids
0	34176	3906	0	6.0	0.0	39	20
1	34176	121	0	6.0	0.0	14	1
2	34176	4356	1	6.0	0.0	18	2
3	34176	2217	0	6.0	0.0	2	1
4	230784	4818	0	NaN	0.0	8	1

,

categories特征构建

```
categories_temp =
user_log.groupby([user_log["user_id"],user_log["seller_id"],user_log["cat_id"]]).
count().reset_index()[["user_id","seller_id","cat_id"]]
categories_temp.head(20)
```

[42]:

,,

	user_id	seller_id	cat_id
0	1	471	389
1	1	739	1252
2	1	925	1023
3	1	1019	992
4	1	1156	1256
5	1	2245	276
6	1	4026	1252
7	1	4177	1252
8	1	4335	389
9	2	420	602
10	2	420	1213
11	2	1179	500
12	2	1544	737
13	2	1679	1130
14	2	1784	420
15	2	1816	276
16	2	1974	737
17	2	1974	1326
18	2	2076	703
19	2	2194	276

,

```
categories_temp1 =
categories_temp.groupby([categories_temp["user_id"],categories_temp["seller_id"]]
).count().reset_index()
categories_temp1.head(10)
```

[43]:

,,,

	user_id	seller_id	cat_id
0	1	471	1
1	1	739	1
2	1	925	1
3	1	1019	1
4	1	1156	1
5	1	2245	1
6	1	4026	1
7	1	4177	1
8	1	4335	1
9	2	420	2

,

```
categories_temp1.rename(columns=
{"seller_id":"merchant_id","cat_id":"categories"},inplace=True)
categories_temp1.head(10)
```

[44]:

,,,

	user_id	merchant_id	categories
0	1	471	1
1	1	739	1
2	1	925	1
3	1	1019	1
4	1	1156	1
5	1	2245	1
6	1	4026	1

	user_id	merchant_id	categories
7	1	4177	1
8	1	4335	1
9	2	420	2

,

```
df_train = pd.merge(df_train,categories_temp1,on=
["user_id","merchant_id"],how="left")
df_train.head(10)
```

[46]:

,,,

	user_id	merchant_id	label	age_range	gender	total_logs	unique_item_ids	categories
0	34176	3906	0	6.0	0.0	39	20	6
1	34176	121	0	6.0	0.0	14	1	1
2	34176	4356	1	6.0	0.0	18	2	1
3	34176	2217	0	6.0	0.0	2	1	1
4	230784	4818	0	NaN	0.0	8	1	1
5	362112	2618	0	4.0	1.0	1	1	1
6	34944	2051	0	5.0	0.0	3	2	1
7	231552	3828	1	5.0	0.0	83	48	15
8	231552	2124	0	5.0	0.0	7	4	1
9	232320	1168	0	4.0	1.0	4	1	1

,

browse_days特征构建

```
browse_days_temp =
user_log.groupby([user_log["user_id"],user_log["seller_id"],user_log["time_stamp"
]]).count().reset_index()[["user_id","seller_id","time_stamp"]]
browse_days_temp.head(10)
```

[48]:

,,

	user_id	seller_id	time_stamp
0	1	471	1111
1	1	739	1018

	user_id	seller_id	time_stamp
2	1	925	1011
3	1	1019	1111
4	1	1156	1111
5	1	2245	1009
6	1	4026	1018
7	1	4026	1021
8	1	4177	1018
9	1	4335	1111

,

```
browse_days_temp1 =
browse_days_temp.groupby([browse_days_temp["user_id"],browse_days_temp["seller_id
"]]).count().reset_index()
browse_days_temp1.head(10)
```

[49]:

,,,

	user_id	seller_id	time_stamp
0	1	471	1
1	1	739	1
2	1	925	1
3	1	1019	1
4	1	1156	1
5	1	2245	1
6	1	4026	2
7	1	4177	1
8	1	4335	1
9	2	420	1

,

```
browse_days_temp1.rename(columns=
{"seller_id":"merchant_id","time_stamp":"browse_days"},inplace=True)
browse_days_temp1.head(10)
```

[50]:

, ,

	user_id	merchant_id	browse_days
0	1	471	1
1	1	739	1
2	1	925	1
3	1	1019	1
4	1	1156	1
5	1	2245	1
6	1	4026	2
7	1	4177	1
8	1	4335	1
9	2	420	1

,

```
df_train = pd.merge(df_train,browse_days_temp1,on=
["user_id","merchant_id"],how="left")
df_train.head(10)
```

[52]:

, ,
, ,

	user_id	merchant_id	label	age_range	gender	total_logs	unique_item_ids	categories	browse_days
0	34176	3906	0	6.0	0.0	39	20	6	9
1	34176	121	0	6.0	0.0	14	1	1	3
2	34176	4356	1	6.0	0.0	18	2	1	2
3	34176	2217	0	6.0	0.0	2	1	1	1
4	230784	4818	0	NaN	0.0	8	1	1	3
5	362112	2618	0	4.0	1.0	1	1	1	1
6	34944	2051	0	5.0	0.0	3	2	1	1
7	231552	3828	1	5.0	0.0	83	48	15	3
8	231552	2124	0	5.0	0.0	7	4	1	1
9	232320	1168	0	4.0	1.0	4	1	1	2

,

one_clicks、shopping_carts、purchase_times、favourite_times特征构建

```
one_clicks_temp =
user_log.groupby([user_log["user_id"],user_log["seller_id"],user_log["action_type
"]]).count().reset_index()[["user_id","seller_id","action_type","item_id"]]
one_clicks_temp.head(10)
```

[54]:

, ,

	user_id	seller_id	action_type	item_id
0	1	471	0	1
1	1	739	0	1
2	1	925	0	3
3	1	925	2	1
4	1	1019	0	10
5	1	1019	2	4
6	1	1156	0	1
7	1	2245	0	5
8	1	4026	0	4
9	1	4026	2	1

,

```
one_clicks_temp.rename(columns=
{"seller_id":"merchant_id","item_id":"times"},inplace=True)
one_clicks_temp.head(10)
```

[56]:

, ,

	user_id	merchant_id	action_type	times
0	1	471	0	1
1	1	739	0	1
2	1	925	0	3
3	1	925	2	1
4	1	1019	0	10

	user_id	merchant_id	action_type	times
5	1	1019	2	4
6	1	1156	0	1
7	1	2245	0	5
8	1	4026	0	4
9	1	4026	2	1

,

```
one_clicks_temp["one_clicks"] = one_clicks_temp["action_type"] == 0
one_clicks_temp["one_clicks"] = one_clicks_temp["one_clicks"] *
one_clicks_temp["times"]
one_clicks_temp.head(10)
```

[59]:

,,
,,,,

	user_id	merchant_id	action_type	times	one_clicks
0	1	471	0	1	1
1	1	739	0	1	1
2	1	925	0	3	3
3	1	925	2	1	0
4	1	1019	0	10	10
5	1	1019	2	4	0
6	1	1156	0	1	1
7	1	2245	0	5	5
8	1	4026	0	4	4
9	1	4026	2	1	0

,

```
one_clicks_temp["shopping_carts"] = one_clicks_temp["action_type"] == 1
one_clicks_temp["shopping_carts"] = one_clicks_temp["shopping_carts"] *
one_clicks_temp["times"]
one_clicks_temp.head(10)
```

[62]:

, ,

	user_id	merchant_id	action_type	times	one_clicks	shopping_carts
0	1	471	0	1	1	0
1	1	739	0	1	1	0
2	1	925	0	3	3	0
3	1	925	2	1	0	0
4	1	1019	0	10	10	0
5	1	1019	2	4	0	0
6	1	1156	0	1	1	0
7	1	2245	0	5	5	0
8	1	4026	0	4	4	0
9	1	4026	2	1	0	0

,

```
one_clicks_temp["purchase_times"] = one_clicks_temp["action_type"] == 2
one_clicks_temp["purchase_times"] = one_clicks_temp["purchase_times"] *
one_clicks_temp["times"]
one_clicks_temp.head(10)
```

[65]:

, ,

	user_id	merchant_id	action_type	times	one_clicks	shopping_carts	purchase_times
0	1	471	0	1	1	0	0
1	1	739	0	1	1	0	0
2	1	925	0	3	3	0	0
3	1	925	2	1	0	0	1
4	1	1019	0	10	10	0	0
5	1	1019	2	4	0	0	4
6	1	1156	0	1	1	0	0
7	1	2245	0	5	5	0	0
8	1	4026	0	4	4	0	0
9	1	4026	2	1	0	0	1

,

```
one_clicks_temp["favourite_times"] = one_clicks_temp["action_type"] == 3
one_clicks_temp["favourite_times"] = one_clicks_temp["favourite_times"] *
one_clicks_temp["times"]
one_clicks_temp.head(10)
```

[68]:

, ,

	user_id	merchant_id	action_type	times	one_clicks	shopping_carts	purchase_times	favourite_times
0	1	471	0	1	1	0	0	0
1	1	739	0	1	1	0	0	0
2	1	925	0	3	3	0	0	0
3	1	925	2	1	0	0	1	0
4	1	1019	0	10	10	0	0	0
5	1	1019	2	4	0	0	4	0
6	1	1156	0	1	1	0	0	0
7	1	2245	0	5	5	0	0	0
8	1	4026	0	4	4	0	0	0
9	1	4026	2	1	0	0	1	0

,

```
four_features =
one_clicks_temp.groupby([one_clicks_temp["user_id"],one_clicks_temp["merchant_id"
]]).sum().reset_index()
four_features.head(10)
```

[71]:

, ,

	user_id	merchant_id	action_type	times	one_clicks	shopping_carts	purchase_times	favourite_times
0	1	471	0	1	1	0	0	0
1	1	739	0	1	1	0	0	0
2	1	925	2	4	3	0	1	0
3	1	1019	2	14	10	0	4	0
4	1	1156	0	1	1	0	0	0
5	1	2245	0	5	5	0	0	0
6	1	4026	2	5	4	0	1	0
7	1	4177	0	1	1	0	0	0
8	1	4335	0	1	1	0	0	0
9	2	420	2	26	23	0	3	0

,

```
four_features = four_features.drop(["action_type","times"], axis=1)
df_train = pd.merge(df_train,four_features,on=
["user_id","merchant_id"],how="left")
df_train.head(10)
```

[74]:

,,
,,
,

	user_id	merchant_id	label	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times
0	34176	3906	0	6.0	0.0	39	20	6	9	36	0	1	2
1	34176	121	0	6.0	0.0	14	1	1	3	13	0	1	0
2	34176	4356	1	6.0	0.0	18	2	1	2	12	0	6	0
3	34176	2217	0	6.0	0.0	2	1	1	1	1	0	1	0
4	230784	4818	0	NaN	0.0	8	1	1	3	7	0	1	0
5	362112	2618	0	4.0	1.0	1	1	1	1	0	0	1	0
6	34944	2051	0	5.0	0.0	3	2	1	1	2	0	1	0
7	231552	3828	1	5.0	0.0	83	48	15	3	78	0	5	0
8	231552	2124	0	5.0	0.0	7	4	1	1	6	0	1	0
9	232320	1168	0	4.0	1.0	4	1	1	2	2	0	1	1

,

建立好的特征的缺失值处理

```
df_train.info()
<class 'pandas.core.frame.DataFrame'>

Int64Index: 260864 entries, 0 to 260863

Data columns (total 13 columns):

user_id            260864 non-null int64

merchant_id        260864 non-null int64

label              260864 non-null int64

age_range          203802 non-null float64

gender             250170 non-null float64

total_logs         260864 non-null int64

unique_item_ids    260864 non-null int64

categories         260864 non-null int64

browse_days        260864 non-null int64

one_clicks         260864 non-null int64

shopping_carts     260864 non-null int64

purchase_times     260864 non-null int64
```

```
favourite_times    260864 non-null int64

dtypes: float64(2), int64(11)

memory usage: 27.9 MB
df_train.isnull().sum(axis=0)
```

[76]:

```
user_id                0
,merchant_id            0
,label                  0
,age_range          57062
,gender             10694
,total_logs             0
,unique_item_ids        0
,categories             0
,browse_days            0
,one_clicks             0
,shopping_carts         0
,purchase_times         0
,favourite_times        0
,dtype: int64
df_train = df_train.fillna(method='ffill')
# 缺失值向前填充
df_train.info()
<class 'pandas.core.frame.DataFrame'>

Int64Index: 260864 entries, 0 to 260863

Data columns (total 13 columns):

user_id            260864 non-null int64

merchant_id        260864 non-null int64

label              260864 non-null int64

age_range          260864 non-null float64

gender             260864 non-null float64

total_logs         260864 non-null int64

unique_item_ids    260864 non-null int64

categories         260864 non-null int64

browse_days        260864 non-null int64

one_clicks         260864 non-null int64

shopping_carts     260864 non-null int64
```

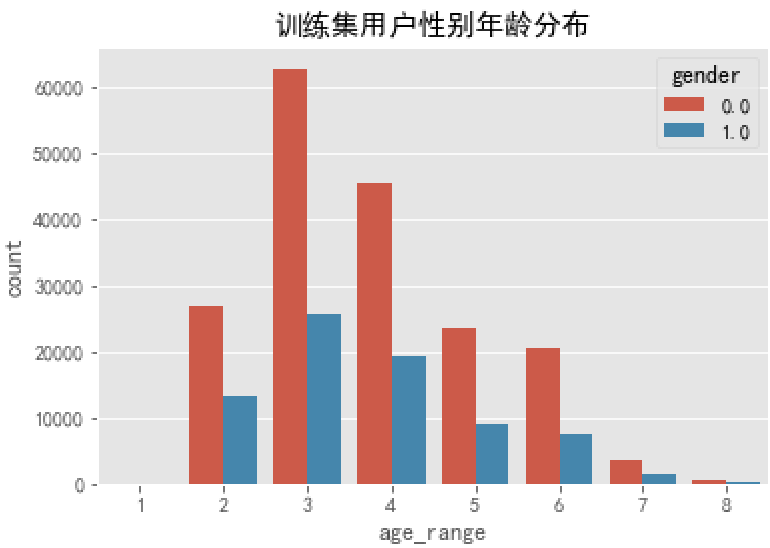
```
purchase_times     260864 non-null int64

favourite_times    260864 non-null int64

dtypes: float64(2), int64(11)

memory usage: 27.9 MB
```

特征可视化

```
plt.style.use('ggplot')
sns.countplot(x = 'age_range', order = [1,2,3,4,5,6,7,8],hue= 'gender',data =
df_train)
plt.title('训练集用户性别年龄分布')
```

[79]:

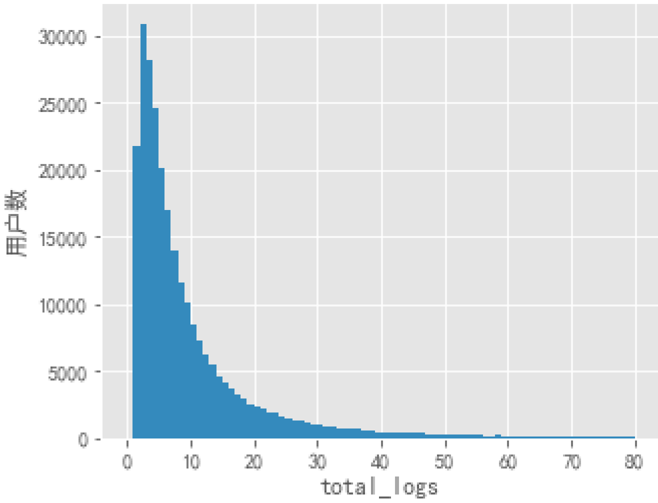
```
Text(0.5, 1.0, '训练集用户性别年龄分布')
```




```
colnm = df_train.columns.tolist()
print(colnm)
plt.figure(figsize = (5, 4))
color = sns.color_palette()

df_train[colnm[5]].hist(range=[0,80],bins = 80,color = color[1])
plt.xlabel(colnm[5],fontsize = 12)
plt.ylabel('用户数')
['user_id', 'merchant_id', 'label', 'age_range', 'gender', 'total_logs',
'unique_item_ids', 'categories', 'browse_days', 'one_clicks', 'shopping_carts',
'purchase_times', 'favourite_times']
```

[80]:

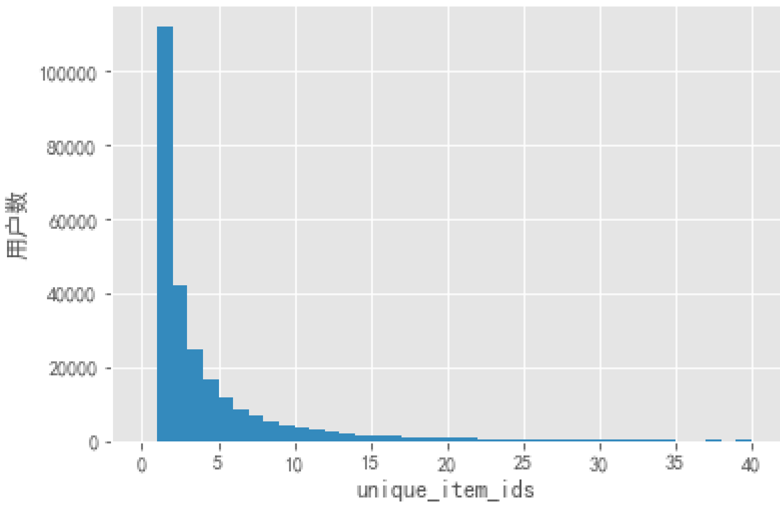
```
Text(0, 0.5, '用户数')
```




```
df_train[colnm[6]].hist(range=[0,40],bins = 40,color = color[1])
plt.xlabel(colnm[6],fontsize = 12)
plt.ylabel('用户数')
```

[81]:

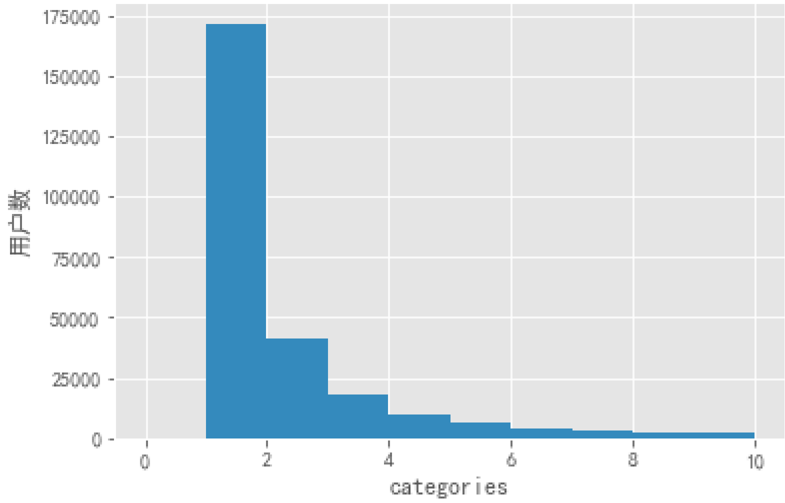
```
Text(0, 0.5, '用户数')
```




```
df_train[colnm[7]].hist(range=[0,10],bins = 10,color = color[1])
plt.xlabel(colnm[7],fontsize = 12)
plt.ylabel('用户数')
```

[82]:

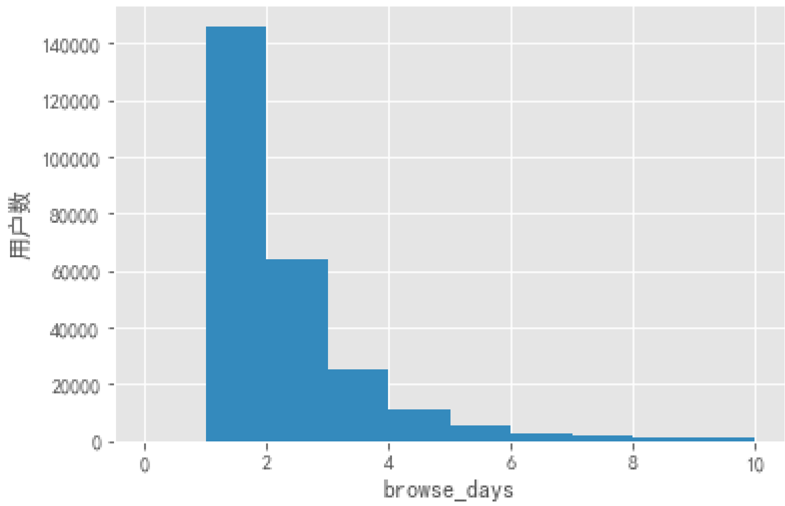
```
Text(0, 0.5, '用户数')
```




```
df_train[colnm[8]].hist(range=[0,10],bins = 10,color = color[1])
plt.xlabel(colnm[8],fontsize = 12)
plt.ylabel('用户数')
```

[83]:

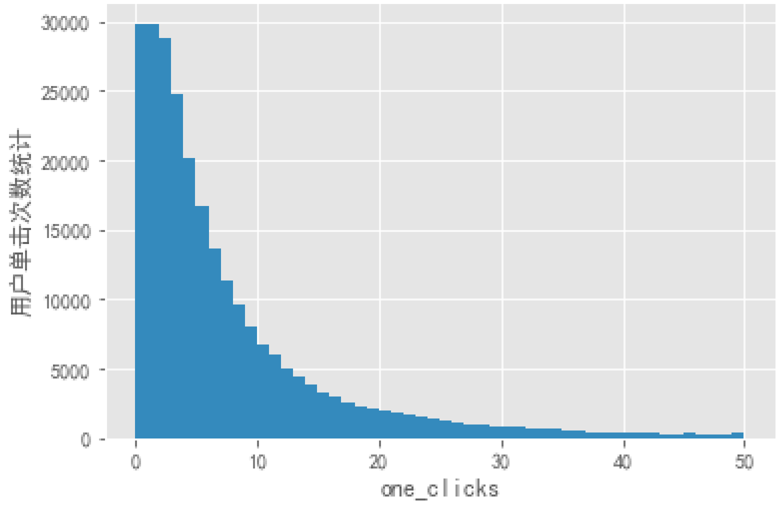
```
Text(0, 0.5, '用户数')
```




```
df_train[colnm[9]].hist(range=[0,50],bins = 50,color = color[1])
plt.xlabel(colnm[9],fontsize = 12)
plt.ylabel('用户单击次数统计')
```

[84]:

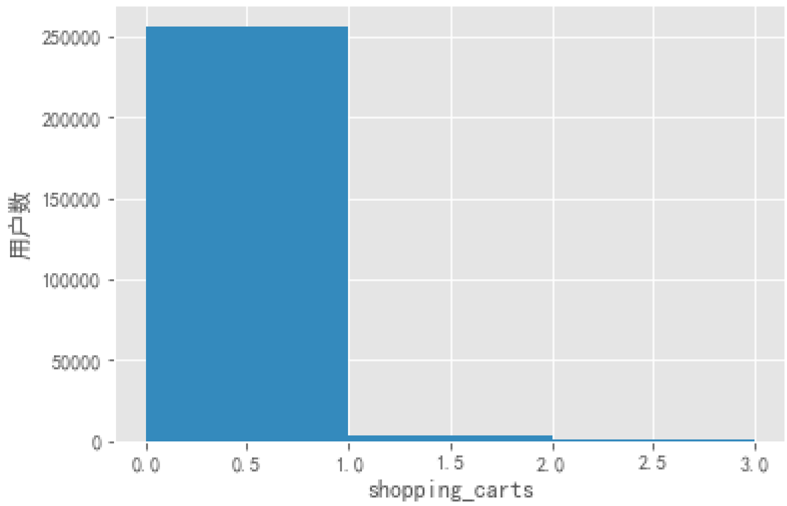
```
Text(0, 0.5, '用户单击次数统计')
```




```
df_train[colnm[10]].hist(range=[0,3],bins = 3,color = color[1])
plt.xlabel(colnm[10],fontsize = 12)
plt.ylabel('用户数')
```

[85]:

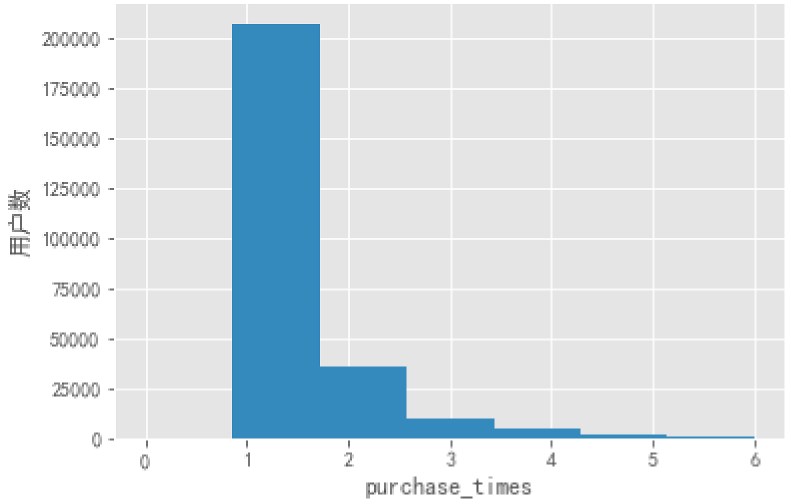
```
Text(0, 0.5, '用户数')
```




```
df_train[colnm[11]].hist(range=[0,6],bins = 7,color = color[1])
plt.xlabel(colnm[11],fontsize = 12)
plt.ylabel("用户数")
```

[86]:

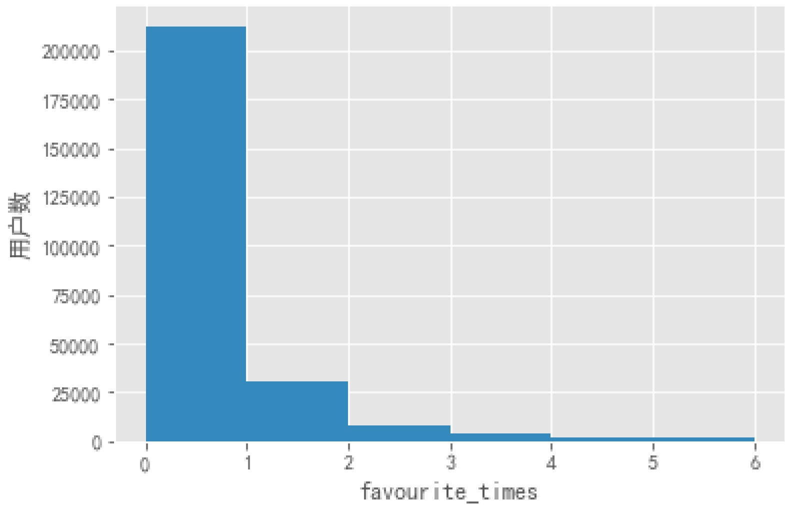
```
Text(0, 0.5, '用户数')
```



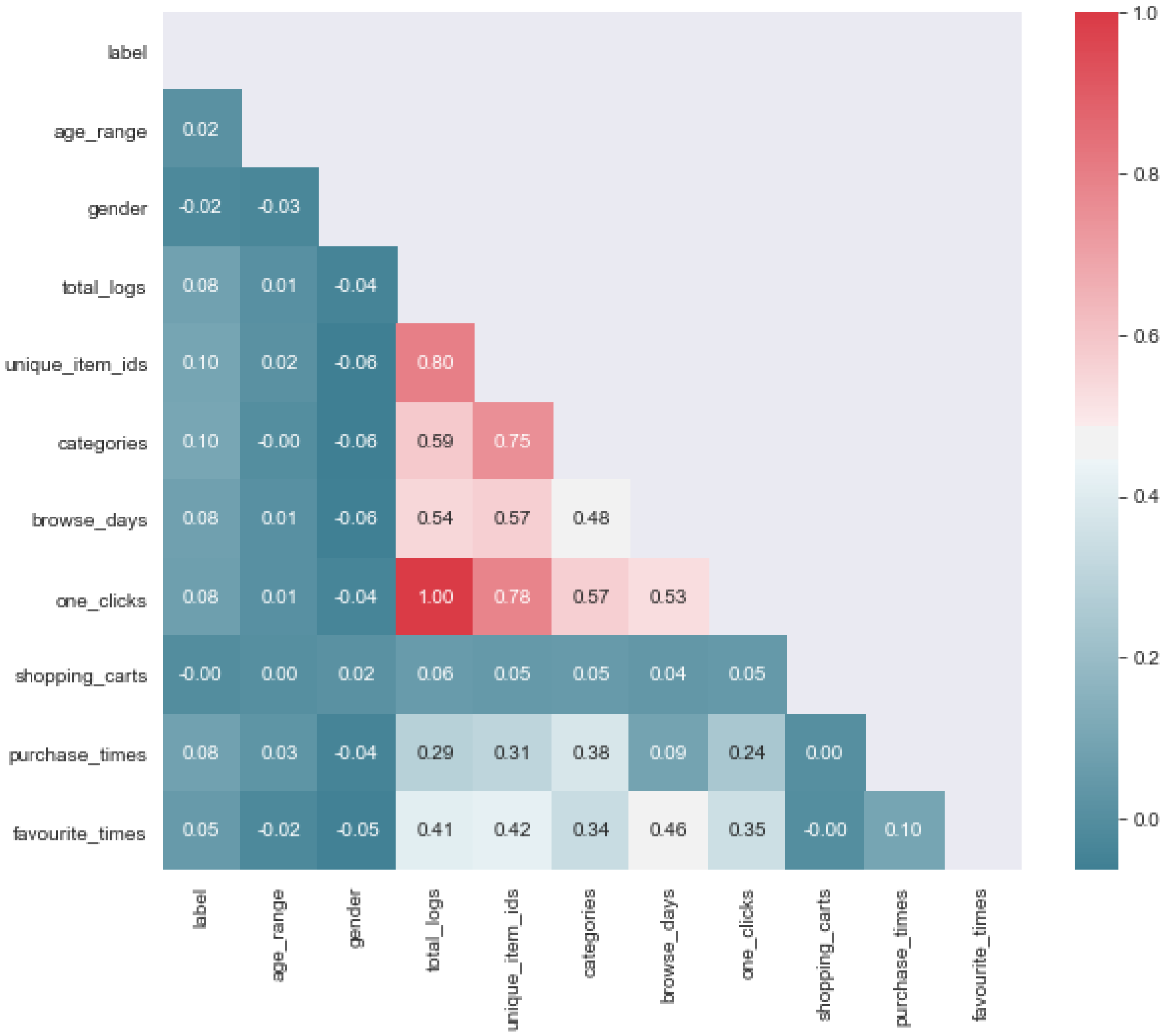

```
df_train[colnm[12]].hist(range=[0,6],bins = 6,color = color[1])
plt.xlabel(colnm[12],fontsize = 12)
plt.ylabel("用户数")
```

[87]:

```
Text(0, 0.5, '用户数')
```




```
sns.set_style("dark")

plt.figure(figsize = (10,8))
colnm = df_train.columns.tolist()[2:13]
mcorr = df_train[colnm].corr()
# np.zero_like的意思就是生成一个和你所给数组a相同shape的全0数组。
mask = np.zeros_like(mcorr, dtype=np.bool)
# np.triu_indices_from()返回方阵的上三角矩阵的索引
mask[np.triu_indices_from(mask)] = True
cmap = sns.diverging_palette(220, 10, as_cmap=True)
g = sns.heatmap(mcorr, mask=mask, cmap=cmap, square=True, annot=True,fmt='0.2f')
# 相关性好像不大，可是日志里确实也没啥可以用的其他特征了啊
```



模型构建与调参

逻辑斯特模型

```
Y = df_train['label']
X = df_train.drop(['user_id','merchant_id','label'],axis = 1)
X.head(10)
```

[89]:

,,

,,

	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times
0	6.0	0.0	39	20	6	9	36	0	1	2
1	6.0	0.0	14	1	1	3	13	0	1	0
2	6.0	0.0	18	2	1	2	12	0	6	0
3	6.0	0.0	2	1	1	1	1	0	1	0
4	6.0	0.0	8	1	1	3	7	0	1	0
5	4.0	1.0	1	1	1	1	0	0	1	0
6	5.0	0.0	3	2	1	1	2	0	1	0
7	5.0	0.0	83	48	15	3	78	0	5	0
8	5.0	0.0	7	4	1	1	6	0	1	0
9	4.0	1.0	4	1	1	2	2	0	1	1

,

```
Y.head(10)
```

[90]:

```
0    0
,1    0
,2    1
,3    0
,4    0
,5    0
,6    0
,7    1
,8    0
,9    0
,Name: label, dtype: int64
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size =
0.25,random_state = 10)
Logit = LogisticRegression(solver='liblinear')
Logit.fit(X_train, y_train)
Predict = Logit.predict(X_test)
Predict_proba = Logit.predict_proba(X_test)
print(Predict[0:20])
print(Predict_proba[:])
Score = accuracy_score(y_test, Predict)
Score
# 一般的准确率验证方法
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[[0.79596485 0.20403515]

 [0.95828116 0.04171884]

 [0.95192756 0.04807244]

 ...

 [0.94051927 0.05948073]

 [0.95677735 0.04322265]

 [0.93242882 0.06757118]]
```

[92]:

```
0.9382053483807654
print("lr.coef_: {}".format(Logit.coef_))
print("lr.intercept_: {}".format(Logit.intercept_))
# 截距与斜率
lr.coef_: [[ 0.0494296  -0.09897898  0.01360686  0.01184834  0.07017633
0.06282481

  -0.01775597 -0.13624657  0.17884728 -0.01123788]]

lr.intercept_: [-3.47137226]
```

```
#初始化逻辑回归算法
LogRegAlg=LogisticRegression(random_state=1,solver='liblinear')
re = LogRegAlg.fit(X,Y)
#使用sklearn库里面的交叉验证函数获取预测准确率分数
scores = model_selection.cross_val_score(LogRegAlg,X,Y,cv=3)
#使用交叉验证分数的平均值作为最终的准确率
print("准确率为: ",scores.mean())
准确率为:  0.9386998590700669
```

K近邻模型

```
# 模型实例化，并将邻居个数设为3
reg = KNeighborsRegressor(n_neighbors=1000)
# 利用训练数据和训练目标值来拟合模型
reg.fit(X_train, y_train)
print("Test set R^2: {:.2f}".format(reg.score(X_test, y_test)))
Test set R^2: 0.01
```

“这一算法对于有很多特 征（几百或更多）的数据集往往效果不好，对于大多数特征的大多数取值都为 0 的数据集 （所谓的稀疏数据集）来说，这一算法的效果尤其不好”我的数据里面零很多，果然预测效果很不好，和闹着玩似的

决策树

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=4,random_state=0)
tree.fit(X_train, y_train)
Predict_proba = tree.predict_proba(X_test)
print(Predict_proba[:])
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
[[0.89760368 0.10239632]

 [0.95837129 0.04162871]

 [0.95837129 0.04162871]

 ...

 [0.91089364 0.08910636]

 [0.95837129 0.04162871]

 [0.93376113 0.06623887]]

Accuracy on training set: 0.939

Accuracy on test set: 0.938
from sklearn.tree import export_graphviz
export_graphviz(tree, out_file="tree.dot", class_names=["0","1"],
feature_names=X.columns.tolist(), impurity=False, filled=True)
# 我们可以利用 tree 模块的 export_graphviz 函数来将树可视化。这个函数会生成一 个 .dot 格式
的文件，这是一种用于保存图形的文本文件格式。
```

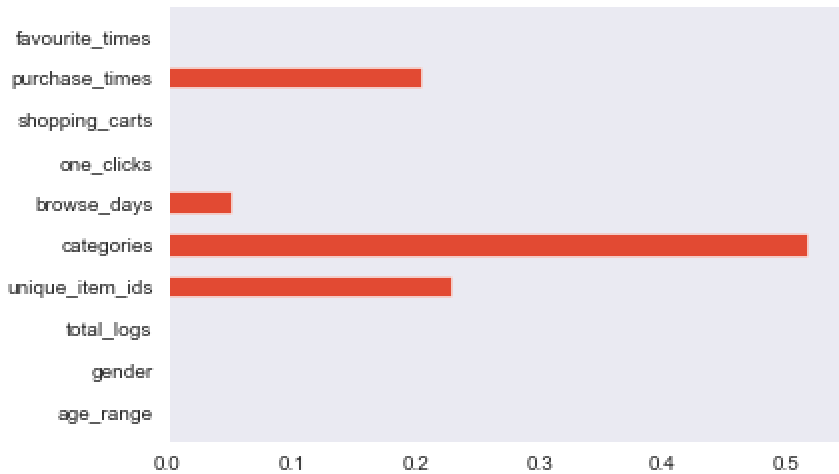
```
# 设置为结点添加颜色 的选项，颜色表示每个结点中的多数类别，同时传入类别名称和特征名称，这样可以对树正确标记
import graphviz
with open("tree.dot") as f:
    dot_graph = f.read()
graphviz.Source(dot_graph)
```

[98]:

```
<graphviz.files.Source at 0x2a5d30dfd68>
print("Feature importances:\n{}".format(tree.feature_importances_))
Feature importances:

[0.         0.         0.         0.22828078 0.51584599 0.05130975

 0.         0.         0.20456347 0.        ]
plt.barh(X.columns.tolist(),height=0.5,width=tree.feature_importances_,align="center")
```

[100]:

```
<BarContainer object of 10 artists>
```



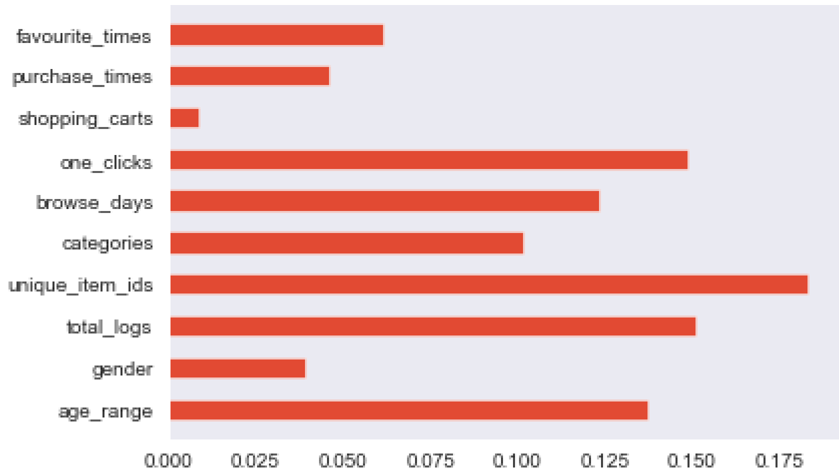
随机森林

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators=10, random_state=2)
forest.fit(X_train, y_train)
Predict_proba = forest.predict_proba(X_test)
print(Predict_proba[:])
print("Accuracy on training set: {:.3f}".format(forest.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(forest.score(X_test, y_test)))
[[0.8        0.2       ]

 [0.95573603 0.04426397]

 [0.95407861 0.04592139]

 ...
```

```
 [1.         0.        ]

 [0.95003622 0.04996378]

 [1.         0.        ]]

Accuracy on training set: 0.959

Accuracy on test set: 0.933
plt.barh(X.columns.tolist(),height=0.5,width=forest.feature_importances_,align="c
enter")
```

[102]:

```
<BarContainer object of 10 artists>
```



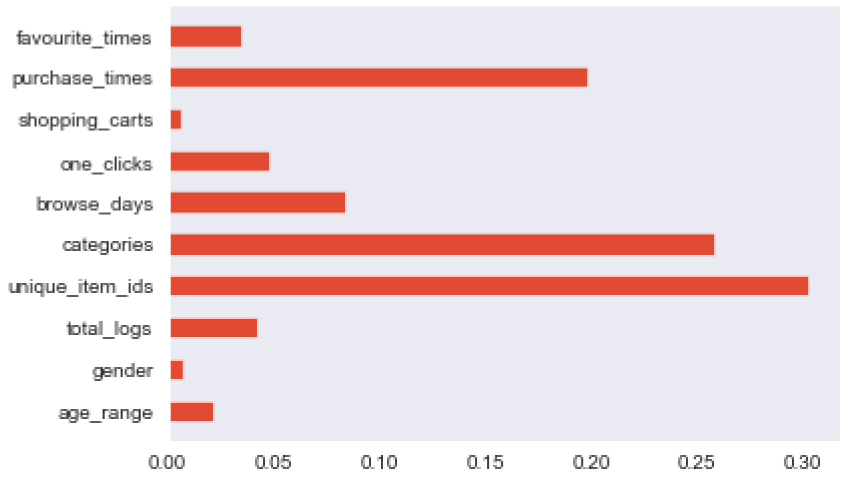
梯度提升回归树

```
from sklearn.ensemble import GradientBoostingClassifier
gbrt = GradientBoostingClassifier(random_state=0)
gbrt.fit(X_train, y_train)
Predict_proba = gbrt.predict_proba(X_test)
print(Predict_proba[:])
print("Accuracy on training set: {:.3f}".format(gbrt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(gbrt.score(X_test, y_test)))
[[0.87042249 0.12957751]

 [0.957535   0.042465  ]

 [0.9548853  0.0451147 ]

 ...

 [0.91902328 0.08097672]

 [0.96209947 0.03790053]

 [0.91587013 0.08412987]]
```

```
Accuracy on training set: 0.939

Accuracy on test set: 0.938
plt.barh(X.columns.tolist(),height=0.5,width=gbrt.feature_importances_,align="center")
```

[104]:

```
<BarContainer object of 10 artists>
```



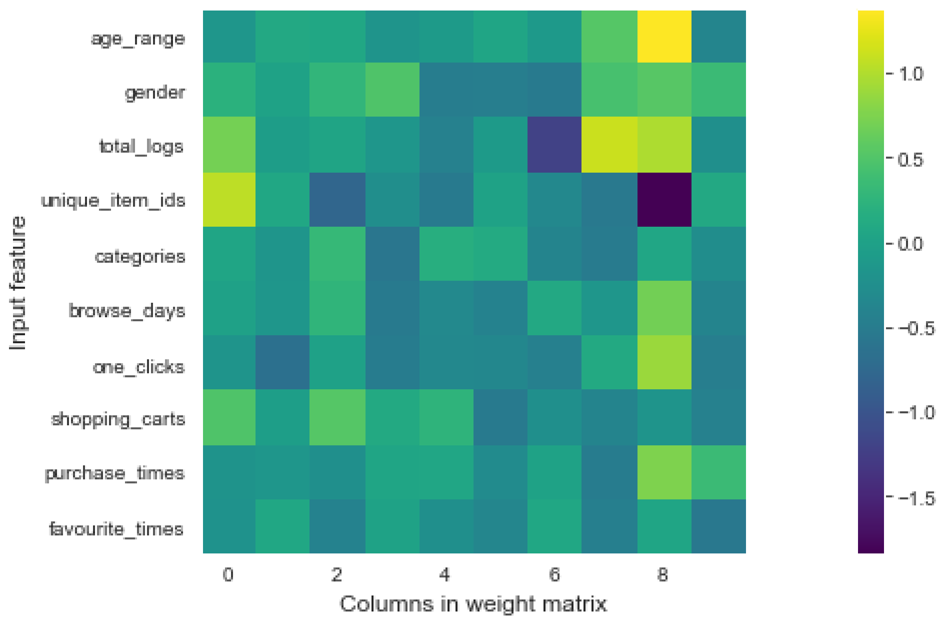
多层感知机

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(solver='lbfgs',
activation='relu',alpha=0.1,random_state=0,hidden_layer_sizes=
[10,10]).fit(X_train, y_train)
Predict = mlp.predict(X_test)
Predict_proba = mlp.predict_proba(X_test)
print(Predict_proba[:])
Score = accuracy_score(y_test, Predict)
print(Score)
[[0.88631107 0.11368893]

 [0.95665074 0.04334926]

 [0.95105044 0.04894956]

 ...

 [0.93326781 0.06673219]

 [0.96103722 0.03896278]

 [0.92282799 0.07717201]]

0.938190014720314
plt.figure(figsize=(20, 5))
plt.imshow(mlp.coefs_[0], interpolation='none', cmap='viridis')
plt.yticks(range(10), X.columns.tolist())
plt.xlabel("Columns in weight matrix")
```

```
plt.ylabel("Input feature")
plt.colorbar()
# 显示了连接输入和第一个隐层之间的权重。图中的行对应 10个输入特征，列对应 10个隐单元。
```

[166]:

```
<matplotlib.colorbar.Colorbar at 0x2a5d561eb00>
```




```
# 原始数据预处理之缩放
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = X_train[X_train.columns.tolist()].astype(float)
X_test = X_test[X_test.columns.tolist()].astype(float)
scaler.fit(X_train)
```

[109]:

```
StandardScaler(copy=True, with_mean=True, with_std=True)
# 变换数据
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
mlp1 = MLPClassifier(solver='lbfgs', random_state=0,hidden_layer_sizes=
[10]).fit(X_train_scaled, y_train)
Predict = mlp1.predict(X_test)
Score = accuracy_score(y_test, Predict)
print(Score)
0.9381286800785084
```

缩放之后的结果也没啥大不了的

实践预测

```
df_test.head()
```

[113]:

, ,

	user_id	merchant_id	prob
0	163968	4605	NaN
1	360576	1581	NaN
2	98688	1964	NaN
3	98688	3645	NaN
4	295296	3361	NaN

,

特征构建

```
df_test = pd.merge(df_test,user_info,on="user_id",how="left")
df_test = pd.merge(df_test,total_logs_temp,on=
["user_id","merchant_id"],how="left")
df_test = pd.merge(df_test,unique_item_ids_temp1,on=
["user_id","merchant_id"],how="left")
df_test = pd.merge(df_test,categories_temp1,on=
["user_id","merchant_id"],how="left")
df_test = pd.merge(df_test,browse_days_temp1,on=
["user_id","merchant_id"],how="left")
df_test = pd.merge(df_test,four_features,on=["user_id","merchant_id"],how="left")
df_test = df_test.fillna(method='bfill')
df_test = df_test.fillna(method='ffill')
# 缺失值向后填充
df_test.head(10)
```

[121]:

, ,
,

	user_id	merchant_id	prob	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times
0	163968	4605	NaN	2.0	0.0	2	1	1	1	1	0	1	0
1	360576	1581	NaN	2.0	0.0	10	9	4	1	5	0	5	0
2	98688	1964	NaN	6.0	0.0	6	1	1	1	5	0	1	0
3	98688	3645	NaN	6.0	0.0	11	1	1	1	10	0	1	0
4	295296	3361	NaN	2.0	1.0	50	8	4	5	47	0	1	2
5	33408	98	NaN	2.0	0.0	11	2	1	4	9	0	1	1
6	230016	1742	NaN	5.0	1.0	13	6	1	1	11	0	2	0
7	164736	598	NaN	3.0	1.0	2	1	1	1	1	0	1	0
8	164736	1963	NaN	3.0	1.0	3	2	1	1	2	0	1	0
9	164736	2634	NaN	3.0	1.0	7	4	3	1	6	0	1	0

,

```
df_test.isnull().sum(axis=0)
```

[122]:

```
user_id                  0
,merchant_id             0
,prob               261477
,age_range               0
,gender                  0
,total_logs              0
,unique_item_ids         0
,categories              0
,browse_days             0
,one_clicks              0
,shopping_carts          0
,purchase_times          0
,favourite_times         0
,dtype: int64
```

模型预测

```
X1 = df_test.drop(['user_id','merchant_id','prob'],axis = 1)
X1.head(10)
```

[124]:

,,,

	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times
0	2.0	0.0	2	1	1	1	1	0	1	0
1	2.0	0.0	10	9	4	1	5	0	5	0
2	6.0	0.0	6	1	1	1	5	0	1	0
3	6.0	0.0	11	1	1	1	10	0	1	0
4	2.0	1.0	50	8	4	5	47	0	1	2
5	2.0	0.0	11	2	1	4	9	0	1	1
6	5.0	1.0	13	6	1	1	11	0	2	0
7	3.0	1.0	2	1	1	1	1	0	1	0
8	3.0	1.0	3	2	1	1	2	0	1	0
9	3.0	1.0	7	4	3	1	6	0	1	0

,

逻辑斯特模型

```
Predict_proba = Logit.predict_proba(X1)
df_test["Logit_prob"] = Predict_proba[:,1]
Predict_proba[0:10]
```

[127]:

```
array([[0.95432087, 0.04567913],
,       [0.87876884, 0.12123116],
,       [0.94574288, 0.05425712],
,       [0.94679761, 0.05320239],
,       [0.9415685 , 0.0584315 ],
,       [0.94633622, 0.05366378],
,       [0.94157686, 0.05842314],
,       [0.95643288, 0.04356712],
,       [0.95611093, 0.04388907],
,       [0.9494901 , 0.0505099 ]])
df_test.head(10)
```

[128]:

,,,
,,,
,,,,,,,,,,,,

	user_id	merchant_id	prob	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times	Logit_prob
0	163968	4605	NaN	2.0	0.0	2	1	1	1	1	0	1	0	0.045679
1	360576	1581	NaN	2.0	0.0	10	9	4	1	5	0	5	0	0.121231
2	98688	1964	NaN	6.0	0.0	6	1	1	1	5	0	1	0	0.054257
3	98688	3645	NaN	6.0	0.0	11	1	1	1	10	0	1	0	0.053202
4	295296	3361	NaN	2.0	1.0	50	8	4	5	47	0	1	2	0.058431
5	33408	98	NaN	2.0	0.0	11	2	1	4	9	0	1	1	0.053664
6	230016	1742	NaN	5.0	1.0	13	6	1	1	11	0	2	0	0.058423
7	164736	598	NaN	3.0	1.0	2	1	1	1	1	0	1	0	0.043567
8	164736	1963	NaN	3.0	1.0	3	2	1	1	2	0	1	0	0.043889
9	164736	2634	NaN	3.0	1.0	7	4	3	1	6	0	1	0	0.050510

,

决策树

```
Predict_proba = tree.predict_proba(X1)
df_test["Tree_prob"] = Predict_proba[:,1]
Predict_proba[0:10]
```

[131]:

```
array([[0.95837129, 0.04162871],
,       [0.89760368, 0.10239632],
,       [0.95837129, 0.04162871],
,       [0.95837129, 0.04162871],
,       [0.93145807, 0.06854193],
,       [0.95232015, 0.04767985],
,       [0.91089364, 0.08910636],
,       [0.95837129, 0.04162871],
,       [0.95232015, 0.04767985],
,       [0.93145807, 0.06854193]])
df_test.head(10)
```

[132]:

,,,
,,,
,,,,,,,,,,,,,,,,,,,,,,,

	user_id	merchant_id	prob	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times	Logit_prob	Tree_prob
0	163968	4605	NaN	2.0	0.0	2	1	1	1	1	0	1	0	0.045679	0.041629
1	360576	1581	NaN	2.0	0.0	10	9	4	1	5	0	5	0	0.121231	0.102396
2	98688	1964	NaN	6.0	0.0	6	1	1	1	5	0	1	0	0.054257	0.041629
3	98688	3645	NaN	6.0	0.0	11	1	1	1	10	0	1	0	0.053202	0.041629
4	295296	3361	NaN	2.0	1.0	50	8	4	5	47	0	1	2	0.058431	0.068542
5	33408	98	NaN	2.0	0.0	11	2	1	4	9	0	1	1	0.053664	0.047680
6	230016	1742	NaN	5.0	1.0	13	6	1	1	11	0	2	0	0.058423	0.089106
7	164736	598	NaN	3.0	1.0	2	1	1	1	1	0	1	0	0.043567	0.041629
8	164736	1963	NaN	3.0	1.0	3	2	1	1	2	0	1	0	0.043889	0.047680
9	164736	2634	NaN	3.0	1.0	7	4	3	1	6	0	1	0	0.050510	0.068542

,

随机森林

```
Predict_proba = forest.predict_proba(X1)
df_test["Forest_prob"] = Predict_proba[:,1]
Predict_proba[0:10]
```

[135]:

```
array([[0.96719674, 0.03280326],
,       [1.        , 0.        ],
,       [0.98963009, 0.01036991],
,       [1.        , 0.        ],
,       [0.9       , 0.1       ],
,       [1.        , 0.        ],
,       [1.        , 0.        ],
,       [0.96004151, 0.03995849],
,       [0.95346438, 0.04653562],
,       [1.        , 0.        ]])
df_test.head(10)
```

[136]:

,,,
,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

	user_id	merchant_id	prob	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times	Logit_prob	Tree_prob	Forest_prob
0	163968	4605	NaN	2.0	0.0	2	1	1	1	1	0	1	0	0.045679	0.041629	0.032803
1	360576	1581	NaN	2.0	0.0	10	9	4	1	5	0	5	0	0.121231	0.102396	0.000000
2	98688	1964	NaN	6.0	0.0	6	1	1	1	5	0	1	0	0.054257	0.041629	0.010370
3	98688	3645	NaN	6.0	0.0	11	1	1	1	10	0	1	0	0.053202	0.041629	0.000000
4	295296	3361	NaN	2.0	1.0	50	8	4	5	47	0	1	2	0.058431	0.068542	0.100000
5	33408	98	NaN	2.0	0.0	11	2	1	4	9	0	1	1	0.053664	0.047680	0.000000
6	230016	1742	NaN	5.0	1.0	13	6	1	1	11	0	2	0	0.058423	0.089106	0.000000
7	164736	598	NaN	3.0	1.0	2	1	1	1	1	0	1	0	0.043567	0.041629	0.039958
8	164736	1963	NaN	3.0	1.0	3	2	1	1	2	0	1	0	0.043889	0.047680	0.046536
9	164736	2634	NaN	3.0	1.0	7	4	3	1	6	0	1	0	0.050510	0.068542	0.000000

,

梯度提升回归树

```
Predict_proba = gbrt.predict_proba(X1)
df_test["Gbrt_prob"] = Predict_proba[:,1]
Predict_proba[0:10]
```

[139]:

```
array([[0.96338469, 0.03661531],
,       [0.89053731, 0.10946269],
,       [0.95615727, 0.04384273],
,       [0.95615727, 0.04384273],
,       [0.92325522, 0.07674478],
,       [0.95456509, 0.04543491],
,       [0.92597124, 0.07402876],
,       [0.96209947, 0.03790053],
,       [0.95641878, 0.04358122],
,       [0.94607038, 0.05392962]])
df_test.head(10)
```

[140]:

,,,
,,,
,,,

	user_id	merchant_id	prob	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times	Logit_prob	Tree_prob	Forest_prob	Gbrt_prob
0	163968	4605	NaN	2.0	0.0	2	1	1	1	1	0	1	0	0.045679	0.041629	0.032803	0.036615
1	360576	1581	NaN	2.0	0.0	10	9	4	1	5	0	5	0	0.121231	0.102396	0.000000	0.109463
2	98688	1964	NaN	6.0	0.0	6	1	1	1	5	0	1	0	0.054257	0.041629	0.010370	0.043843
3	98688	3645	NaN	6.0	0.0	11	1	1	1	10	0	1	0	0.053202	0.041629	0.000000	0.043843
4	295296	3361	NaN	2.0	1.0	50	8	4	5	47	0	1	2	0.058431	0.068542	0.100000	0.076745
5	33408	98	NaN	2.0	0.0	11	2	1	4	9	0	1	1	0.053664	0.047680	0.000000	0.045435
6	230016	1742	NaN	5.0	1.0	13	6	1	1	11	0	2	0	0.058423	0.089106	0.000000	0.074029
7	164736	598	NaN	3.0	1.0	2	1	1	1	1	0	1	0	0.043567	0.041629	0.039958	0.037901
8	164736	1963	NaN	3.0	1.0	3	2	1	1	2	0	1	0	0.043889	0.047680	0.046536	0.043581
9	164736	2634	NaN	3.0	1.0	7	4	3	1	6	0	1	0	0.050510	0.068542	0.000000	0.053930

,

多层感知机

```
Predict_proba = mlp.predict_proba(X1)
df_test["mlp_prob"] = Predict_proba[:,1]
Predict_proba[0:10]
```

[169]:

```
array([[0.95341203, 0.04658797],
,       [0.91441547, 0.08558453],
,       [0.95840793, 0.04159207],
,       [0.96310161, 0.03689839],
,       [0.90702839, 0.09297161],
,       [0.9461217 , 0.0538783 ],
,       [0.93960292, 0.06039708],
,       [0.95786441, 0.04213559],
,       [0.94904894, 0.05095106],
,       [0.93775615, 0.06224385]])
df_test.head(10)
```

[170]:

,,,
,,,
,,

	user_id	merchant_id	prob	age_range	gender	total_logs	unique_item_ids	categories	browse_days	one_clicks	shopping_carts	purchase_times	favourite_times	Logit_prob	Tree_prob	Forest_prob	Gbrt_prob	mlp_prob
0	163968	4605	NaN	2.0	0.0	2	1	1	1	1	0	1	0	0.045679	0.041629	0.032803	0.036615	0.046588
1	360576	1581	NaN	2.0	0.0	10	9	4	1	5	0	5	0	0.121231	0.102396	0.000000	0.109463	0.085585
2	98688	1964	NaN	6.0	0.0	6	1	1	1	5	0	1	0	0.054257	0.041629	0.010370	0.043843	0.041592
3	98688	3645	NaN	6.0	0.0	11	1	1	1	10	0	1	0	0.053202	0.041629	0.000000	0.043843	0.036898
4	295296	3361	NaN	2.0	1.0	50	8	4	5	47	0	1	2	0.058431	0.068542	0.100000	0.076745	0.092972
5	33408	98	NaN	2.0	0.0	11	2	1	4	9	0	1	1	0.053664	0.047680	0.000000	0.045435	0.053878
6	230016	1742	NaN	5.0	1.0	13	6	1	1	11	0	2	0	0.058423	0.089106	0.000000	0.074029	0.060397
7	164736	598	NaN	3.0	1.0	2	1	1	1	1	0	1	0	0.043567	0.041629	0.039958	0.037901	0.042136
8	164736	1963	NaN	3.0	1.0	3	2	1	1	2	0	1	0	0.043889	0.047680	0.046536	0.043581	0.050951
9	164736	2634	NaN	3.0	1.0	7	4	3	1	6	0	1	0	0.050510	0.068542	0.000000	0.053930	0.062244

,

结果保存

```
choose = ["user_id","merchant_id","mlp_prob"]
res = df_test[choose]
res.rename(columns={"mlp_prob":"prob"},inplace=True)
print(res.head(10))
res.to_csv(path_or_buf =
r"C:\Users\gaohao\Downloads\data_format1\prediction.csv",index = False)
D:\anaconda\lib\site-packages\pandas\core\frame.py:3781: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame



See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy

  return super(DataFrame, self).rename(**kwargs)
   user_id  merchant_id      prob

0   163968         4605  0.035843

1   360576         1581  0.097485

2    98688         1964  0.042821

3    98688         3645  0.047452

4   295296         3361  0.067267

5    33408           98  0.038221

6   230016         1742  0.076726

7   164736          598  0.036325

8   164736         1963  0.044482

9   164736         2634  0.050063
```