Sensors Data Project Report

Ziwei Wang

Project overview

Sensorsdata is a leading China-based and rapidly growing big data company and interested in User-behavior Analytics and Conversion Prediction. To help explore this question, they have provided log data containing tens of thousands of log information of Sensorsdata main webpage for a week.

Github address: https://github.com/will-zw-wang/Sensors_Data

Dataset description

Data is given as txt file whereas data is in JSON format. Dataset contains cache log information of Sensorsdata main webpage for a week, including actions on leaving the webpage, click a button, send verification code, apply for account, etc.
Please refer to Log Description for detailed description:
https://github.com/will-zw-wang/Sensors_Data/tree/master/log_description

A. Data exploration and cleaning

1. Data exploration

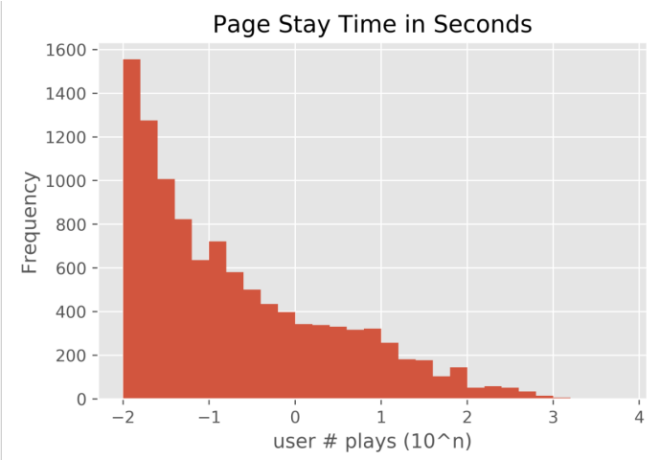
1.1 Select identifier

'distinct_id', hash: change from browser and cookie id to protect user privacy; 11708 distinct values
'nocache': might be simplified identifier. The number of distinct nocache is 65659. The frequency is too low.
'ip': 9870 distinct values, each ip address can have multiple users

Conclusion:
'distinct_id' is a better identifier， it is random， and can be used for unbiased experiments test split

2 Numeric variables

2.1 page_stayTime



Insights:

The plot shows that most of user stayed in the page for less than 1 second (10^-1), and the 75 percentile 'page_stayTime' value is 0.226 second, which means most of the records with 'page_stayTime' value did not mean user really stay in the page, they might leave when the page was loading.

By calculation, we know only 11.41 % of the people who clicked the functional pages stayed more than 3 seconds

Create features "page_stayTime": Sum up the total stay time by pages

index_leave_stayTime	demo_leave_stayTime	about_leave_stayTime	courses_leave_stayTime	courses_play_leave_stayTime
0.006	0.000	0.0	0.0	0.0
0.000	0.000	0.0	0.0	0.0
0.000	0.000	0.0	0.0	0.0
0.000	0.000	0.0	0.0	0.0
0.000	0.021	0.0	0.0	0.0

3 Categorical variables

3.1 event

```
df['event'].value_counts()
```

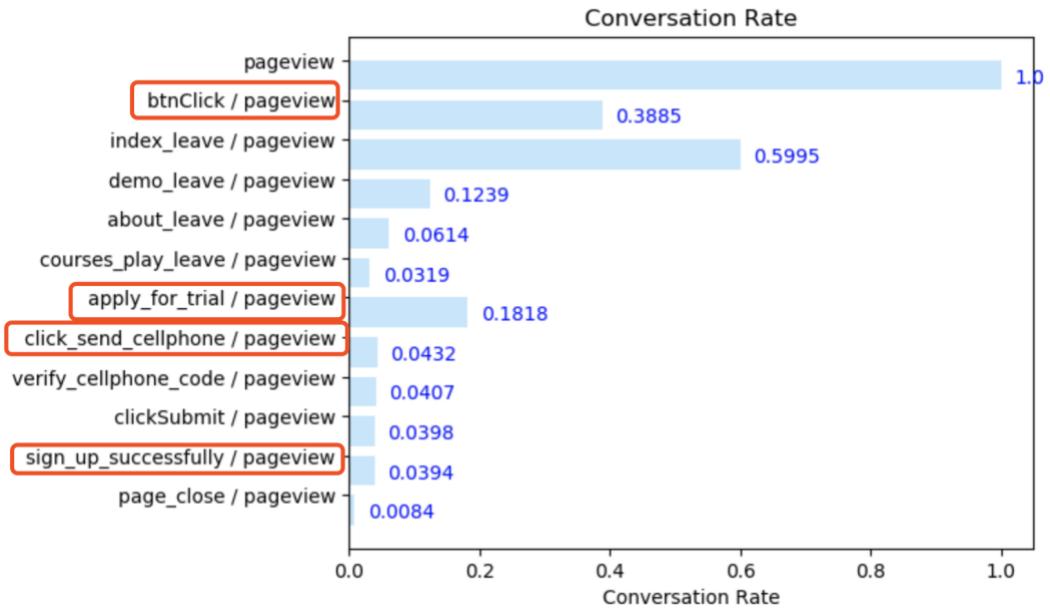
```
$pageview                32620
btnClick                 13866
index_leave              10394
demo_leave                3411
about_leave               1032
courses_leave              906
formSubmit                 791
courses_play_leave         747
click_send_cellphone       600
verify_cellphone_code      563
clickSubmit                513
page_close                 230
Name: event, dtype: int64
```

Funnel Analysis:

page view -> btnClick -> click_send_cellphone -> verify_cellphone_code -> clickSubmit -> formSubmit

We Calculated conversation between funnel sections and plot:

event	distinct_dist_id_count	$pageview_dist_id_count	Conversation Rate
page_close	97	11587	0.0084
sign_up_successfully	457	11587	0.0394
clickSubmit	461	11587	0.0398
verify_cellphone_code	472	11587	0.0407
click_send_cellphone	501	11587	0.0432
apply_for_trial	2106	11587	0.1818
courses_play_leave	370	11587	0.0319
about_leave	712	11587	0.0614
demo_leave	1436	11587	0.1239
index_leave	6946	11587	0.5995
btnClick	4501	11587	0.3885
$pageview	11587	11587	1.0000



Note:

Here we define '**signup**' with the action 'click_send_cellphone', which means 'dist_id' attempts to sign up an account.
We define '**signup successfully**' with 'isSuccess' property of 'formSubmit' is 'True'.
We define '**apply for trial**' with 'name' property of 'btnClick' is 'request'.

From the plot above, we notice that:

'**button_click_rate':** 38.85 % of the people who viewed the webpage and clicked a button.
'**apply_for_trial_rate**': 18.18 % of the people who viewed the webpage and clicked the 'request' button.
'**signup_to_apply_for_trial_rate':** 23.79 % of the people who clicked the 'request' button and clicked 'send_cellphone' attempting to sign up.
'**signup_rate**': 4.32 % of the people who viewed the webpage and clicked 'send_cellphone' attempting to sign up.
'**successfully_signup_rate**': 91.22 % of the people who clicked 'send_cellphone' attempting to signup did successfully registered.

Insights：

1. 'button_click_rate' is only 38.85%, most users do not click buttons on pages.
We should improve the layout of our pages, modify the wording or color of our buttons to make our pages more attractive.

2. 'apply_for_trial_rate' is only 18.18%, most users do not click 'request' buttons on pages.
We should polish our service description, modify the wording or color of our 'request' buttons to make our pages more attractive.

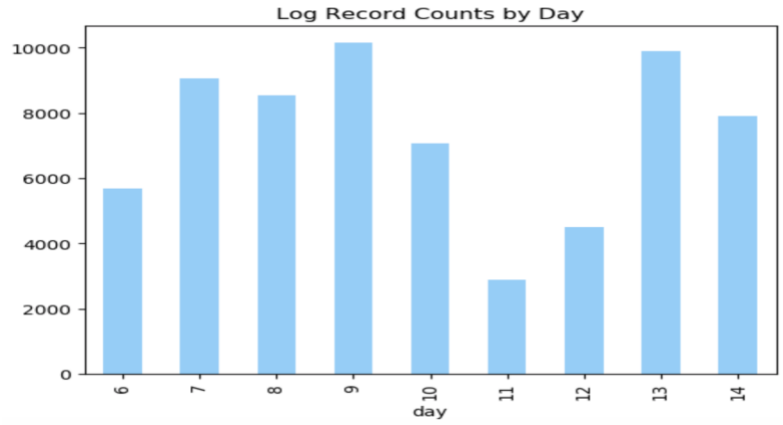
3. 'signup_to_apply_for_trial_rate' is only 23.79%, most users who clicked 'request' button at the beginning did not apply when they were asked to provide phone number, perhaps the users care about personal privacy.
Providing other registration options, like e-mail or social network accounts, may be a good choice.

4. 'signup_rate' is only 4.32%, most users who viewed the webpage did not attempt to sign up, the same idea with 'apply_for_trial_rate'.

5. 'successfully_signup_rate' is 91.22%, we lost nearly 9% of users who attempting to sign up.
We should try to improve the efficiency of our sign up process, provide other registration options.

3.2 day

We plotted log record counts by Day:



Insights：

2017-03-11 and 2017-03-12 are weekends and have the least number of counts, perhaps users visit our website more for work purposes.

create feature 'weekend'： With values: '1' for weekend, '0' and weekdays

```
df['weekend'].value_counts(dropna = False)
```

```
0    58300
1     7373
Name: weekend, dtype: int64
```

3.3 title

```
df[df['event'] == '$pageview']["title"].value_counts(dropna=False)
```

```
神策数据 | Sensors Data - 国内领先的用户行为分析产品                    13700
神策分析|Sensors Analytics-帮你实现数据驱动－demo                     4148
介绍 · Sensors Analytics 使用手册                                3542
神策分析|Sensors Analytics-帮你实现数据驱动－产品                     2455
神策分析|Sensors Analytics-帮你实现数据驱动－B轮融资发布会              1899
神策分析|Sensors Analytics-帮你实现数据驱动－关于                     1183
神策分析|Sensors Analytics-帮你实现数据驱动-视频列表                   1129
                                                              ...
C SDK · Sensors Analytics 使用手册                                1
3 神策分析|Sensors Analytics 帮你实现数据驱动－demo                    1
```

Insights:

Users visit our 'demo' page, 'user manual' page and 'product' page most frequently. If we plan to improve the layout of our pages, these three pages should have high priority.

3.4 latest_referrer_host

```
df[df['event'] == '$pageview']['latest_referrer_host'].value_counts(dropna=False)
```

```
NaN                    13771
www.baidu.com          13146
m.baidu.com              983
36kr.com                 694
www.sogou.com            687
www.google.com.hk        416
www.google.com           346
                        ...
www.xfz.cn                 1
```

Insights:

Most of empty 'latest_referrer_host' are directly from sensordata website.
The other users were referred mostly from 'baidu', '36kr', 'sogou' and 'google', especially 'baidu' which contributed times of referred users than the other hosts.
Thus, if we want to run marketing campaign, 'baidu' should be allocated more budget to.

create feature 'latest_referrer_host_bin': With values: 'sensordata', 'baidu' and 'others'

```
df.latest_referrer_host_bin.value_counts(dropna = False)
```

```
baidu         31378
sensordata    25299
others         8996
Name: latest_referrer_host_bin, dtype: int64
```

3.5 ip

```
df['ip'].value_counts(dropna=False)
```

```
113.208.116.250    1399
113.208.112.126     673
113.208.118.30      392
122.233.41.29       367
106.38.73.242       247
118.194.240.141     244
61.190.32.52        233
                   ...
140.207.21.106        1
```

Insights:

The same ip might be used by several users.

We map the ip to specific cities:

```
df['city'].value_counts()
```

```
Beijing          16288
Shanghai          5396
Shenzhen          4361
Guangzhou         3667
Hangzhou          2759
Haidian           1947
Chengdu           1750
                  ...
Paris                1
Wafangdian           1
```

Insights:

We notice most of the users come from 'Beijing', 'Shanghai', 'Shenzhen' and 'Guangzhou'.

create feature 'city_bin': With values 'Beijing', 'Shanghai', 'Shenzhen', 'Guangzhou' and 'others'

```
df['city_bin'].value_counts(dropna = False)
```

```
others       35961
Beijing      16288
Shanghai      5396
Shenzhen      4361
Guangzhou     3667
Name: city_bin, dtype: int64
```

3.6 name

```
df['name'].value_counts()
```

```
request            2967
demo               2783
document           2067
product            1942
b-round            1245
about               925
blog                803
                   ...
果*                    1
3**                    1
```

Insights:

We notice the mostly clicked buttons are 'request', 'demo', 'document' and 'product'.
If we plan to improve the layout of our pages, the pages these buttons link to should have high priority.

B. Feature Engineering

1. Feature generation

1.1 pages_total_stayTime

Create feautres 'pages_total_stayTime': Sum up the total stay time by pages per 'dist_id'

dist_id	index_page_total_stayTime	demo_page_total_stayTime	about_page_total_stayTime	courses_page_total_stayTime	cou
00007ef910b6c9911f1b89d01a09aa3fc862f4a9	0.000	0.0	0.0	0.000	
000a216b72eff19bd0d5e17b9e676dd6ad9a38ac	921.142	0.0	0.0	0.000	
000c46a27ef69fa22b56d253a9c72773338a1686	10.384	0.0	0.0	1.812	
000ed1dcd942969b458c5b308937c6389c08f999	0.031	0.0	0.0	0.000	
00111feff544ef5280a4c7064a362a9ea59c9389	0.000	0.0	0.0	0.000	

1.2 click_counts

Create feautres 'click_counts': Sum up the click times per 'dist_id'

dist_id	click_counts
00007ef910b6c9911f1b89d01a09aa3fc862f4a9	1
000c46a27ef69fa22b56d253a9c72773338a1686	6
000ed1dcd942969b458c5b308937c6389c08f999	2
00111feff544ef5280a4c7064a362a9ea59c9389	1
0011f5066b1c62717255852fdb15a0473a5c2b19	3

1.3 pages_viewed_counts

Create feautres 'pages_viewed_counts: Sum up the page viewed count per 'dist_id'

dist_id	pages_leave_counts
000a216b72eff19bd0d5e17b9e676dd6ad9a38ac	1
000c46a27ef69fa22b56d253a9c72773338a1686	7
000ed1dcd942969b458c5b308937c6389c08f999	1
0011f5066b1c62717255852fdb15a0473a5c2b19	1
0012ea1b517e6959354abaa6954711054ec831b9	1

2. Transform data and define label

2.1 Converting categorical variables into dummy variables

```
df_dummies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11708 entries, 00007ef910b6c9911f1b89d01a09aa3fc862f4a9 to
Data columns (total 16 columns):
latest_referrer_host_bin_baidu          11708 non-null uint8
latest_referrer_host_bin_others         11708 non-null uint8
latest_referrer_host_bin_sensordata     11708 non-null uint8
latest_utm_source_bin_baidu             11708 non-null uint8
latest_utm_source_bin_others            11708 non-null uint8
latest_utm_source_bin_sensordata        11708 non-null uint8
browser_bin_chrome                      11708 non-null uint8
browser_bin_others                      11708 non-null uint8
city_bin_Beijing                        11708 non-null uint8
city_bin_Guangzhou                      11708 non-null uint8
city_bin_Shanghai                       11708 non-null uint8
city_bin_Shenzhen                       11708 non-null uint8
city_bin_others                         11708 non-null uint8
model_bin_mac                           11708 non-null uint8
model_bin_others                        11708 non-null uint8
model_bin_pc                            11708 non-null uint8
dtypes: uint8(16)
memory usage: 594.4+ KB
```

2.2 define label

We define 'signup' with the action 'click_send_cellphone', which means 'dist_id' attempts to sign up an account.

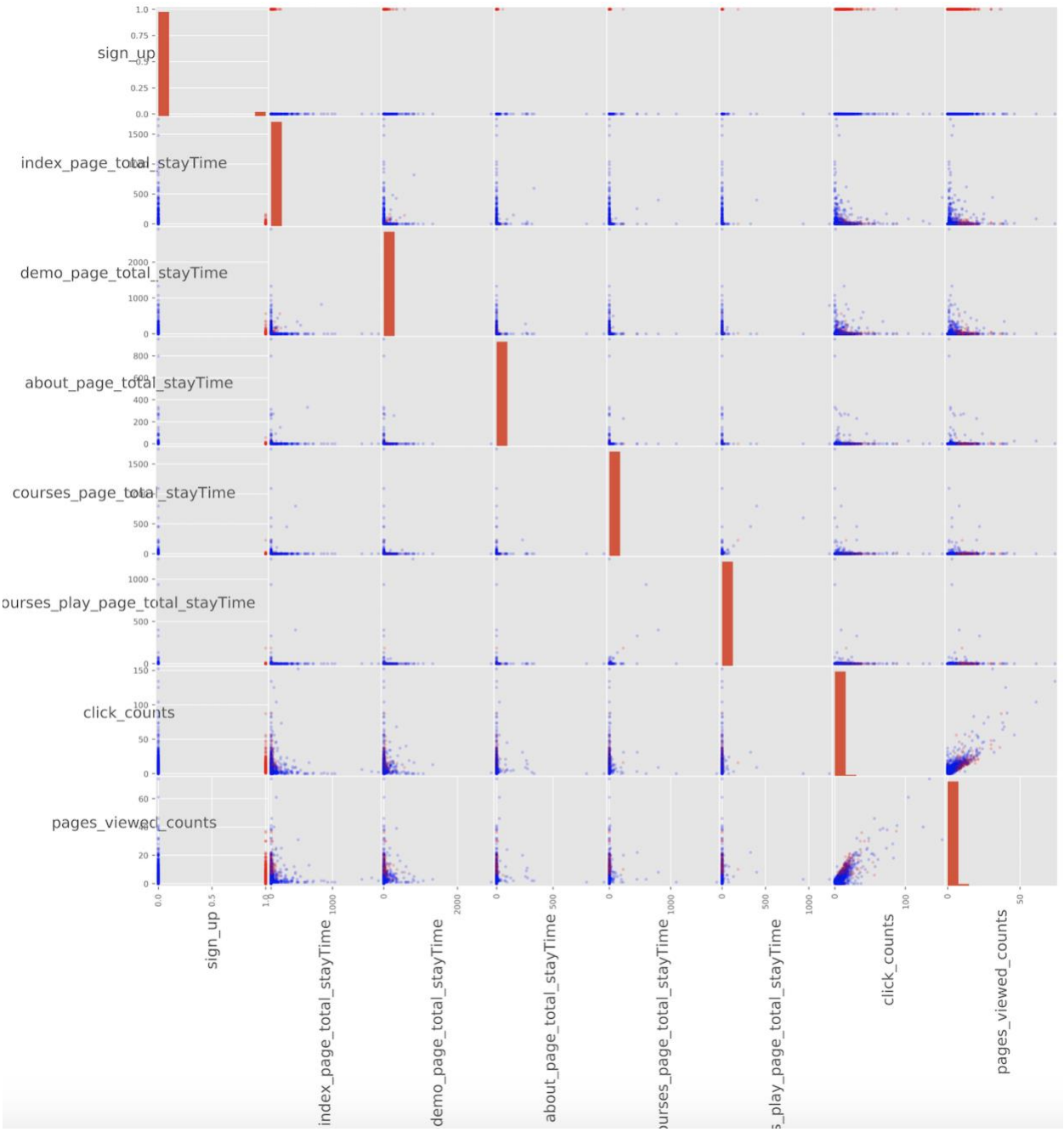
```
df_cleaned['sign_up'].value_counts()
```

```
0.0    11207
1.0      501
Name: sign_up, dtype: int64
```

Sign up rate is only 4.28 %

3. EDA with label

3.1 Scatter_matrix



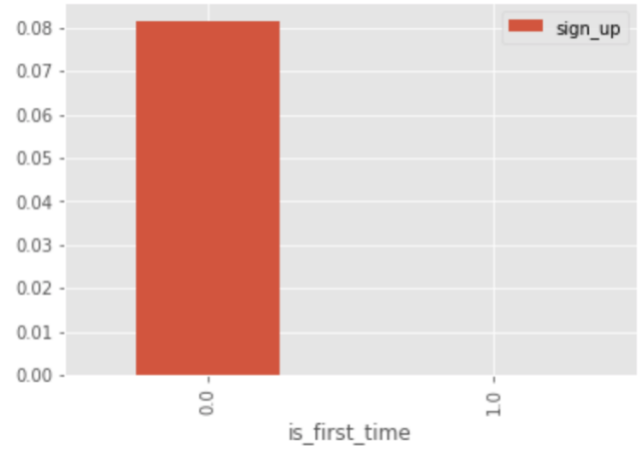
Note: 'red' for 'sign_up', 'blue' for not_sign_up.

Insights:

We notice '**click_counts**' and '**pages_viewed_counts**' are highly correlated with 'sign_up'.

3.2 Explore sign up rate split by features

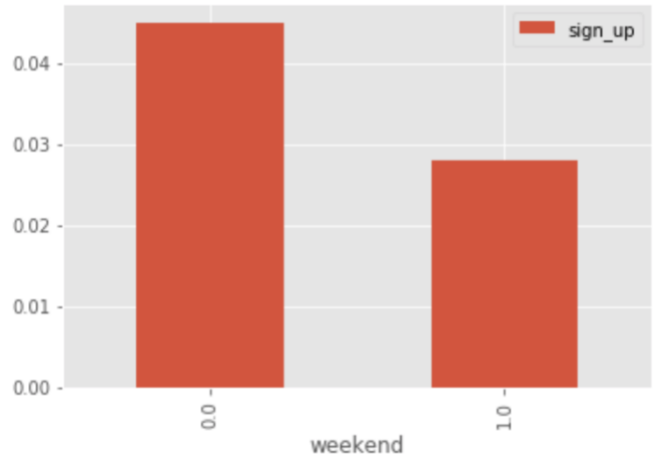
3.2.1 'is_first_time'



Insights:

All users with 'is_first_time' value '1' did not sign_up, which means highly interested users will come to register another time.

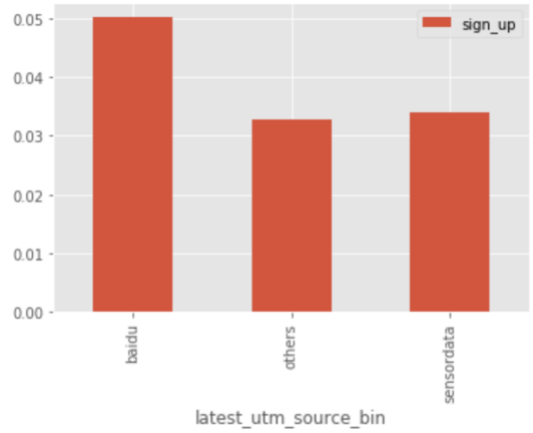
3.2.2 'weekend'



Insights:

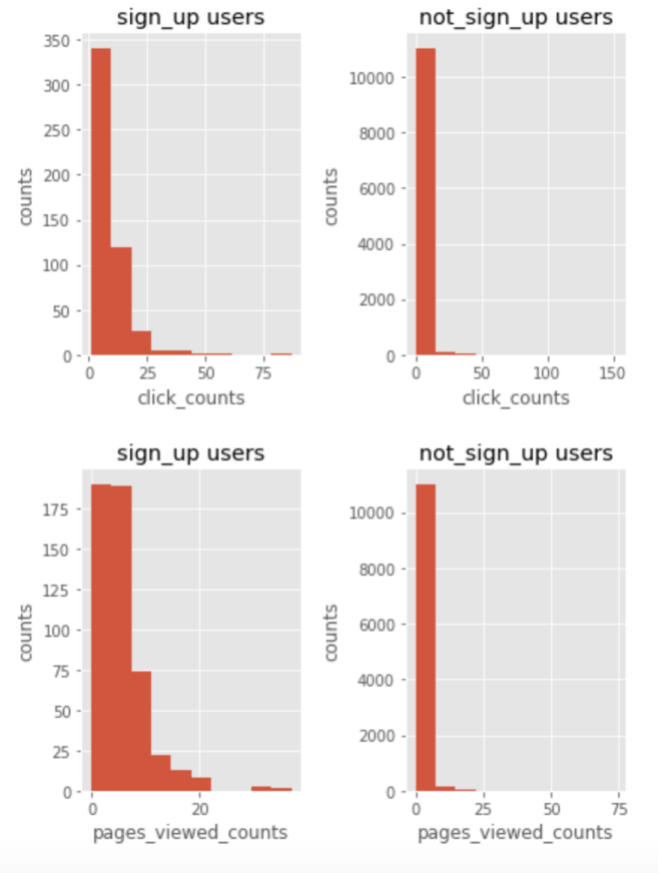
Sign_up rate of 'weekend' is obviously lower than that of 'weekdays', perhaps users visit our website more for work purposes.

3.2.3 ‘latest_utm_source_bin’



Insights:
Sign_up rate of 'latest_utm_source_bin' with value 'baidu' is obviously higher than those of others, if we want to run marketing campaign, 'baidu' should be allocated more budget to.

3.2.4 value distribution of numerical features of ‘sign_up_users’ vs that of ‘not_sign_up_users’



Insights:

As what we observed in the scatter_matrix plotted above, distribution of 'click_counts' of 'sign_up_users' and 'not_sign_up_users' are obviously different, the same idea with 'pages_viewed_counts'

C. Models and Insights

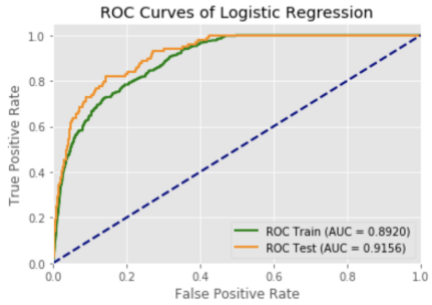
1. Models comparison and reasoning

1.1 Logistic Regression

As we are taking a classification task, we first try Logistic Regression.

The performance of our model as below:

metrics	train	test
AUC	0.891985	0.915643
Accuracy	0.956118	0.959863
Precision	0.415094	0.631579
Recall	0.054726	0.121212
f1-score	0.096703	0.203390

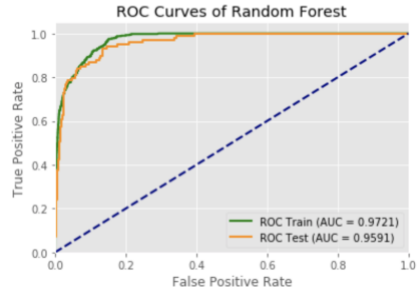


AUC of test data is **0.9156** with **Logistic Regression**, we will try to improve model performance with **Random Forest.**

1.2 Random Forest

The performance of our model as below:

metrics	train	test
AUC	0.972108	0.959114
Accuracy	0.971813	0.966268
Precision	0.852041	0.700000
Recall	0.415423	0.353535
f1-score	0.558528	0.469799

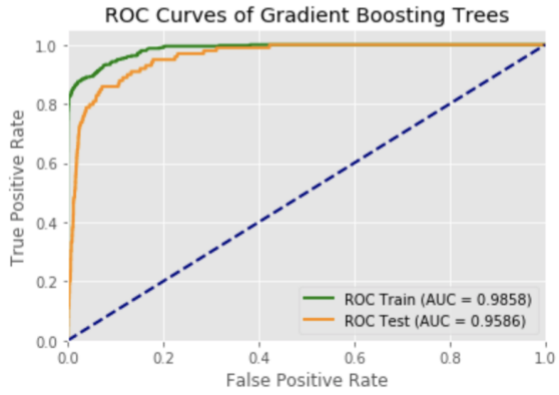


AUC of test data is **0.9591** with **Random Forest**, better than that of **Logistic Regression with 0.9156**, because there are feature interaction and non-linearity relationship between features and target in our data set, trees algorithms can deal with these problems while logistic regression cannot.

Then we tried to further improve model performance with **Gradient Boosting Trees**, because in general, Gradient Boosting Trees can perform better than Random Forest, because it additionally tries to find optimal linear combination of trees (assume final model is the weighted sum of predictions of individual trees) in relation to given train data. This extra tuning may lead to more predictive power.

1.3 Gradient Boosting Trees

metrics	train	test
AUC	0.985780	0.958578
Accuracy	0.989537	0.965414
Precision	0.957831	0.615385
Recall	0.791045	0.484848
f1-score	0.866485	0.542373



AUC of test data is **0.9586** with **Gradient Boosting Trees**, is close to that of **Random Forest** with **0.9591**, means our Random forest has already performed greatly in this dataset and hard for Gradient Boosting Trees to perform better.

Thus, we chose Random Forest as our preferred model here.

Then we tried to implement Random Forest with bootstraps for imbalance dataset to check if balance dataset can improve model performance?

1.4 Random Forest with bootstraps for imbalance dataset

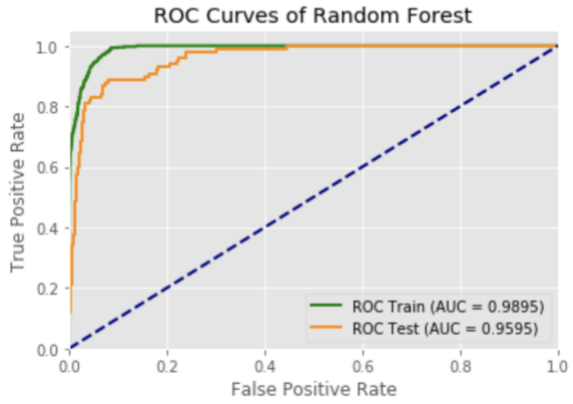
Before bootstraps:

```
0.0    8964
1.0     402
Name: target, dtype: int64
```

After bootstraps:

```
Random over-sampling:
1.0    8964
0.0    8964
Name: target, dtype: int64
```

metrics	train	test
AUC	0.989517	0.959470
Accuracy	0.950747	0.922289
Precision	0.934416	0.338521
Recall	0.969545	0.878788
f1-score	0.951656	0.488764



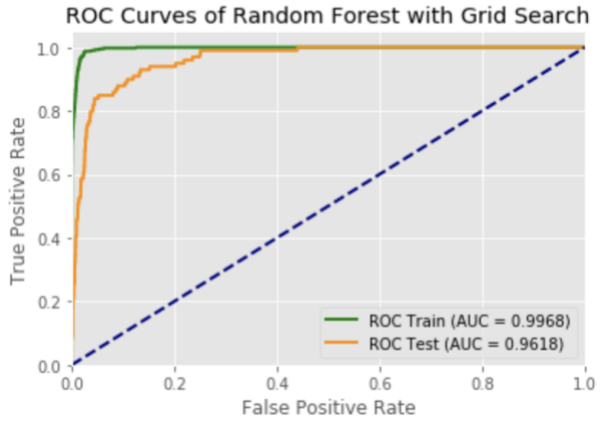
AUC of test data is 0.9595 with **'Random Forest with bootstraps for imbalance dataset'**, is close to that of **Random Forest with the imbalance dataset** with **0.9591**, means our Random forest has already performed greatly with the imbalance dataset and do not need to balance the dataset.

Next, we will try HyperParameter Tuning with Grid Search for Random Forest with the original imbalance dataset, to figure out whether we can do better.

2. HyperParameter Tuning with Grid Search

2.1 Random Forest HyperParameter Tuning with Grid Search

metrics	train	test
AUC	0.996814	0.961769
Accuracy	0.986440	0.966695
Precision	0.982456	0.666667
Recall	0.696517	0.424242
f1-score	0.815138	0.518519



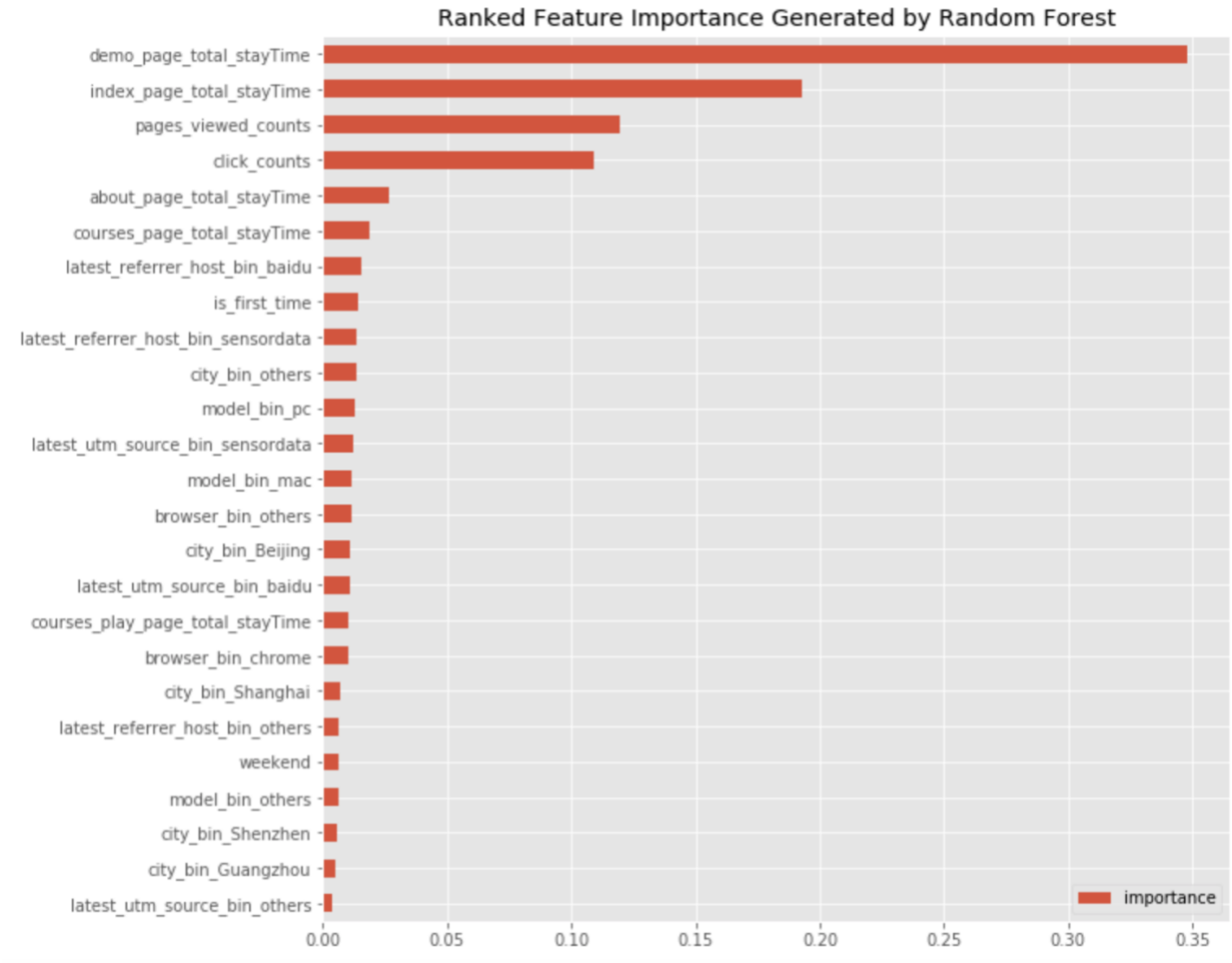
AUC of test data is **0.9617** with **Random Forest HyperParameter Tuning with Grid Search**, is slightly better than that of previous **Random Forest** with **0.9591**, we select this model to explore the features importance to get some insights

We select this model to explore the features importance to get some insights

3. Explore features importance to get insights

3.1 Top 10 features analysis

The plot below shows the **ranked feature importance** generated by Random Forest:



As we can see, the **top 10 features** are:

1. 'demo_page_total_stayTime': the longer time a user spent in 'demo' page is, the more likely the user will sign up. And its feature importance is larger than those of 'index_page_total_stayTime', 'about_page_total_stayTime' and courses_page_total_stayTime', which shows user is more likely to visit our 'demo' page than the others.

2. 'index_page_total_stayTime': the same idea with 'demo_page_total_stayTime', and its feature importance is larger than those of 'about_page_total_stayTime' and courses_page_total_stayTime'.

3. 'pages_viewed_counts': the more pages a user views, the more likely the user will sign up.

4. 'click_counts': the more click a user performs, the more likely the user will sign up.

5. 'about_page_total_stayTime': the same idea with 'demo_page_total_stayTime'.

6. 'courses_page_total_stayTime': the same idea with 'demo_page_total_stayTime'.

7. 'latest_referrer_host_bin_baidu': Users referred by 'baidu' are more likely to sign up than users referred by other channels.

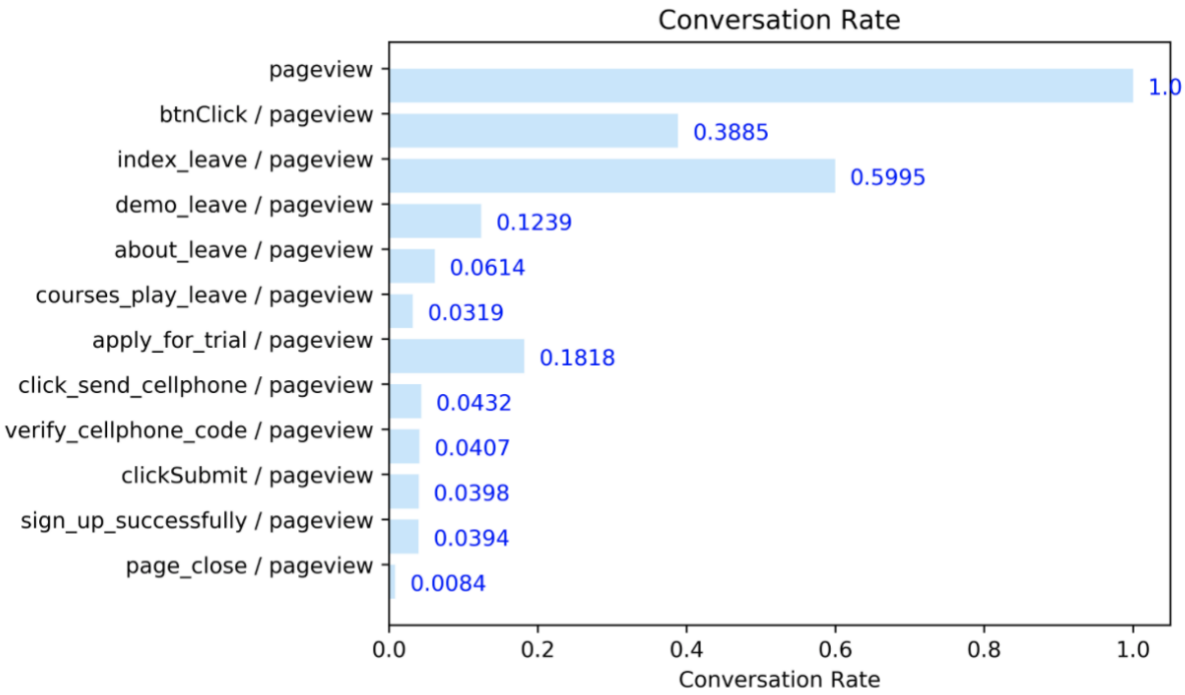
8. 'is_first_time': As we notice in the previous 'feature exploration' part, all users with 'is_first_time' value '1' did not sign up, which means highly interested users will come to register another time, we should give users more times to contract with us to make them sign up.

9. 'latest_referrer_host_bin_sensordata': Most of users visit our pages from sensordata website without any 'referrer_host', which means most of campaigns except 'baidu' have no positive effects.

10. 'city_bin_others': the feature importance of 'city_bin_others' is larger than those of other 'city_bin' values, like 'city_bin_Beijing', 'city_bin_Shanghai', which means signup rate may be random among cities.

3.2 Insights

a. **Funnel Analysis**



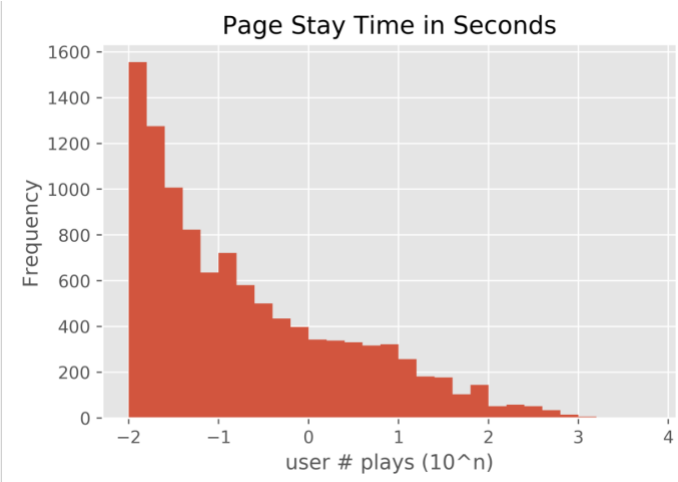
Note:

Here we define '**signup**' with the action 'click_send_cellphone', which means 'dist_id' attempts to sign up an account.
We define '**signup successfully**' with 'isSuccess' property of 'formSubmit' is 'True'.
We define '**apply for trial**' with 'name' property of 'btnClick' is 'request'.

As we mentioned in the previous data exploration section:

1. '**button_click_rate**' is only 38.85%, most users do not click buttons on pages, perhaps the wording or color of our buttons are not attractive enough.
2. '**apply_for_trial_rate**' is only 18.18%, most users do not click 'request' buttons on pages, perhaps our service description is not attractive enough.
3. '**signup_to_apply_for_trial_rate**' is only 23.79%, most users who clicked 'request' button at the beginning did not apply when they were asked to provide phone number, perhaps the users care about personal privacy.
4. '**signup_rate**' is only 4.32%, most users who viewed the webpage did not attempt to sign up, the same idea with 'apply_for_trial_rate'.
5. '**successfully_signup_rate**' is 91.22%, we lost nearly 9% of users who attempting to signup, perhaps the efficiency of our sign up process still need to improve.

b. **page_stayTime Analysis**



As we mentioned in the previous data exploration section:

The plot shows that most of user stayed in the page for less than 1 second(10^-1), and the 75 percentile 'page_stayTime' value is 0.226 second, which means most of the users might left when the page was loading. Only 11.41 % of the people who clicked the functional pages stayed more than 3 seconds

By analyzing the top 10 features, funnel and page_stayTime, we have some insights:
1. We should improve our page quality, given the high feature importance of 'pages_total_stayTime' and very low percent of users stayed in our pages more than 3 seconds.
Like: hire web UX designer to improve the layout of our pages, especially 'demo' page and 'index' page, modify the wording or color of our buttons, polish our service description.

2. We should consider providing other registration options and improve efficiency of our sign up process, given low 'signup_to_apply_for_trial_rate' and low 'successfully_signup_rate'.
Like: allow users to sign up with e-mail or social network accounts.

3. We should make adjustment to product promotion and campaign strategy, given most our campaigns have no significant effect.
Note: as we don't know the current product promotion and campaign strategy of sensordata, we analyze two different scenarios as below:
Scenarios one: if sensordata had already invested lots of money to product promotion and campaign, it should adjust the investment allocation and invest more budget in 'Baidu', which has relatively better performance than the medias.
Scenarios two: if sensordata did not invest much product promotion and campaign before, it should allocate more budget in this area, 'baidu', '36kr', 'sogou' and 'google' would be good choices, especially 'baidu' which contributed times of referred users than the other hosts.

3.3 Next step

Besides the insights mentioned above, I think there are aspects we can further dive deep, like:

1. Detailed analysis on different medium and campaign contributions, try to figure out which channels to invest and how to allocate the budgets.

2. Detailed analysis on user behavior on specific pages, try to figure out which part of the page users pay most attention to, improve the content of interest and redesign the sections that are not valued.

3. It's also important to track performance over time. If we have more data, we can see whether we're improving or not, by comparing funnels for each month.