Tina Gholami
Report Summer Camp 1400
tina.gholamy229@gmail.com

Divar

سوال اول)

- اگر تمام رکوردهای اولین کوئری را که برای فرد با شناسه دستگاه "Ed9EADRZRXCHcrcEP_Hnkg" است را مشاهده کنیم می‌بینیم که هنگامی که فرد فقط آگهی‌ها را load می‌کند دارای post_page_offset و tokens غیرتهی است ولی post_token و post_index_in_post_list برایش تهی هستند چون مربوط به حالت click کردن هستند، و برعکس (منطقی است چون click کردن و load شدن از هم مستقل اند که در قسمت بعدی نشان داده شده است.). وجود این خاصیت شبه باینری بین دو حالت click کردن و load شدن که باعث ظهور NaN value ها در داده می‌شود، ممکن است به هنگام تحلیل داده در مراحل بعدی (استفاده از یادگیری ماشین برای تحلیل داده) مشکلاتی را ایجاد کند. باید حواسمان به این مورد باشد و بدانیم که برای تحلیل، در هر لحظه، دو تا از این چهار ستون دارای مقادیر NaN هستند، هرچند که منطقی است و به قولی، داده‌ی گم شده یا missing value نیستند. طبیعی و منطقی است که وقتی کاربر لیستی را لود می‌کند ولی روی هیچ کدام از آگهی‌ها کلیک نمی‌کند از بین این چهار ستون، دوتای آن‌ها که مربوط به لود شدند هستند، غیرتهی باشند و دوتای دیگر که مربوط به کلیک کردن هستند، NaN باشند و ما نباید آن‌ها را به بهانه اینکه missing value هستند حذف کنیم. بلکه باید با روش دیگری (متناسب با هدفی که از تحلیل داده داریم) با این مقادیرِ نال، برخورد کنیم.

	action	created_at	source_event_id	device_id	post_page_offset	tokens	post_index_in_post_list	post_token
0	load_post_page	1609545001150	1575558c-a702-46ef-8e18-bc5cef761473	Ed9EADRZRXCHcrcEP_Hnkg	26.0	[wXvP3enu, wXvHXzUS, wXvPHXVe, wXvPHRs3, wXvH3...	NaN	NaN
1926	load_post_page	1609544971861	1575558c-a702-46ef-8e18-bc5cef761473	Ed9EADRZRXCHcrcEP_Hnkg	23.0	[wXvX2f3c, wXvnn5z9, wXvjnsRM, wXWH48kW, wXvjm...	NaN	NaN
2201	load_post_page	1609544991175	1575558c-a702-46ef-8e18-bc5cef761473	Ed9EADRZRXCHcrcEP_Hnkg	25.0	[wXvT3kZU, wXvTHI6r, wXvb3ZIZ, wXv7VLK4, wXvDX...	NaN	NaN
2221	load_post_page	1609544845034	1575558c-a702-46ef-8e18-bc5cef761473	Ed9EADRZRXCHcrcEP_Hnkg	19.0	[wXv_X7RE, wXwChV01, wXqnUkds, wXv7nhmH, wXv7n...	NaN	NaN
2226	load_post_page	1609544820280	1575558c-a702-46ef-8e18-bc5cef761473	Ed9EADRZRXCHcrcEP_Hnkg	17.0	[wXaDiMYq, wXjb4S3Z, wXvP4Alc, wXvDInVQ, wXvHI...	NaN	NaN

Figure 1, NaN Values in the data

- مورد دیگری که در دیتا وجود دارد مربوط به حالت کلیک کردن است. زمانی که یوزر روی یک آگهی کلیک می‌کند (click = 'click_post')، ستون‌های post_index_in_post_list و post_token هردو باید باهم غیرتهی باشند زیرا این دو ستون هردو وابسته به اکشن کلیک کردن هستند. در نتیجه، هردو باید به تعداد یکسانی داده‌ی غیرتهی بداشته باشند. اما همان‌طور که در تصویر زیر دیده می‌شود، این طور نیست! و ستون post_token، به اشتباه یک داده کمتر از ستون post_index_in_post_list دارد:

```
data[data['action'] == 'click_post'].count()
```

```
action                  75796
created_at              75796
source_event_id         75796
device_id               74818
post_page_offset            0
tokens                      0
post_index_in_post_list 75796
post_token              75795
dtype: int64
```

Figure 2, The correct number of clicks is 75796 times

```
print('The number of actions in clicking where feature "post_index_in_post_list" is not null is {0}' .format(data[data['actic
print('The number of actions in clicking where feature "post_token" is not null is {0} ' .format(data[data['action'] == 'clic

#This is weird! Because when the user clicks on an add (action = 'click_post')
#then features "post_index_in_post_list" and "post_token" should be both non-NaN since they are concurrent and dependant.
#And therefore, they should have the same number of non-NaN values.
#But as seen bellow, "post_token" has one less non-NaN value.
```

```
The number of actions in clicking where feature "post_index_in_post_list" is not null is 75796
The number of actions in clicking where feature "post_token" is not null is 75795
```

Figure 3, The inconsistency in the data for feature "post_toekn" compared to feature "post_index_in_post_lis", related to action clicking

از طرفی، همین مورد برای دو ستونِ مربوط به اکشن لود شدن (action = 'load_post_page') هم چک شد. یعنی به هنگام لود شدن لیستِ آگهی‌ها، انتظار می‌رود ستون‌های post_page_offset و tokens هردو باهم باید غیرتهی باشند. طبق عکس زیر، برای حالت لود شدن، این انتظار به درستی برآورده شد:

```
print('The number of actions in loading where feature "post_index_in_post_list" is not null is {0}' .format(data[data['action
print('The number of actions in loading where feature "post_token" is not null is {0} ' .format(data[data['action'] == 'load_

#When a list is loaded (action = 'load_post_page'), both features "post_page_offset" and "tokens" are non-NaN.
#And logically they are showing the same number of entries since they are concurrent and dependant.
```

```
The number of actions in loading where feature "post_index_in_post_list" is not null is 35287
The number of actions in loading where feature "post_token" is not null is 35287
```

Figure 4, The consistency in the data for features "tokens" and "post_page_offset", related to action loading

همان‌طور که می‌بینیم، 75796 تا از کوئری‌ها مربوط به اکشن کلیک کردن است و 35287 تا از کوئری‌ها مربوط به اکشن لود شدن است.

از طرفی استقلال ستون‌های مربوط به اکشن لود شدن (post_page_offset و tokens) را از ستون‌های مربوط به اکشن کلیک کردن (post_index_in_post_list و post_token) چک می‌کنیم. به این صورت که اگر ستون‌های مربوط به حالت لود شدنِ آگهی‌ها، غیرتهی بودند، در آن صورت ستون‌های مربوط به حالت کلیک کردن باید تهی باشند، و برعکس.

```
print('The number of loads "post_page_offset" while already clicked: {0}' .format(data[data['action'] == 'click_post']['post_
print('The number of loads "tokens" while already clicked: {0}' .format(data[data['action'] == 'click_post']['tokens'].count(
print('The number of clicks "post_token" while already loaded: {0}' .format(data[data['action'] == 'load_post_page']['post_to
print('The number of clicks "post_index_in_post_list" while already loaded: {0}' .format(data[data['action'] == 'load_post_pa


#The 0 outputs show that when a list is ONLY loaded, then it is independatn from when an add is ONLY clicked on.
#It shows the two actions loading and clicking are distinctive and independatnt from each other.
#Thus, there is no overlap between them.
```

```
The number of loads "post_page_offset" while already clicked: 0
The number of loads "tokens" while already clicked: 0
The number of clicks "post_token" while already loaded: 0
The number of clicks "post_index_in_post_list" while already loaded: 0
```

Figure 5, The independency of action loading from action clicking

همان‌طور که می‌بینیم، صحت استقلال این دو اکشن نشان داده شد.

همچنین، چک میکنیم که حاصل جمع تعداد کوئری‌های مربوط به اکشن لود شدن و اکشن کلیک کردن برابر با تعداد کل کوئری‌ها در دیتاست باشد، چون در قسمت بالا نشان دادیم که این دو عمل و ستون‌های مربوطه شان از هم مستقل اند و overlap بین آن‌ها وجود ندارد.

```
num_loadings = 35287
num_clicks = 75796
len_data = len(data)

num_loadings + num_clicks == len_data

#This was expected too! The first number (35287) as explained above is showing the number of ONLY Loadings.
#And the second number (75796) is showing the number of ONLY clickings.
#And since in the above cell we concluded that loadings and clickings are independant,
#then summing them up should equall to the length of the data, which it did since he output below is "True".
#Also, just to highlight it again, one data is wrongly considered as NaN for fetaure "post_token".
#Therefore, the correct number of clicks in general is 75796, NOT 75795!!
#Hence we have 35287 number of loadings and 75796 number of cliscks!
```

```
True
```

Figure 6, The sum of the number of clicked queries and the number of loaded queries equals the number of total queries, showing there is no overlap between these two actions in the dataset

- از طرفی، تعدادی داده‌ی missing value در دیتاست دیده می‌شود. (منظورم چهار ستون آخر در دیتاست نیست. چون در بالا توضیح دادم که لودینگ‌ها و کلیک‌ها از هم مستقل اند. پس هر وقت ستون‌های مربوط به لودینگ (یعنی دو ستون post_page_offset و tokens) غیرتهی باشند، ناچاراً ستون‌های مربوط به کلیک (یعنی ستون‌های post_index_in_post_list و post_token)، تهی خواهند بود.) برای مثال، در ستون device_id حدود 1487 داده‌ی NaN وجود دارد که برای الگوریتم‌های ماشین لرنینگ باید آن‌ها را با روش‌های مناسب هندل کرد.

```
#Checking for missing values:

data.isnull().sum()
```

```
action                      0
created_at                  0
source_event_id             0
device_id                1487
post_page_offset        75796
tokens                  75796
post_index_in_post_list 35287
post_token              35288
dtype: int64
```

Figure 7, The missing values in the dataset features

- نوع داده‌ای که در ستون created_at ذخیره شده است برحسب timestamp است ولی خیلی واضح نیست. مثلا در اولین instance از داده، مقدار 1609545001150 ثبت شده است ولی با نگاه کردن به این عدد، به راحتی نمی‌توان متوجه شد چه تاریخی را نشان می‌دهد. از طرفی unix هم نمی‌تواند باشد زیرا unix timestamp فقط 10 رقم دارد ولی اعدادی که در دیتاست ثبت شده اند 13 رقمی اند. من این مدل نشان دادن تاریخ و زمان را در دیتابیس‌ها هم ندیده‌ام، پس شاید یک قرارداد یا انکودینگ باشد که تیم دیوار برای ذخیره‌سازی زمان استفاده می‌کند.

- در ستون tokens، طبق توضیحاتی که در دستور پروژه آمده بود، انتظار می‌رفت که type همه‌ی داده‌ها از جنس string باشد ولی طبق عکس زیر، مشخص می‌شود که بعضی از داده‌ها برای این ستون به صورت float در قالب لیست دخیره شده اند:

```
In [79]:
#Just to make sure there are 24 ads in each loaded list:

flag = 0

for i in range(0, len(data['tokens'])):
    if len(data['tokens'][i].split(',')) != 24:
        flag += 1
flag
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-79-56644fe83be3> in <module>
      4
      5 for i in range(0, len(data['tokens'])):
----> 6     if len(data['tokens'][i].split(',')) != 24:
      7         flag += 1
      8 flag

AttributeError: 'float' object has no attribute 'split'
```

Figure 8, The miss-types in feature "tokens"

پس چک می‌کنیم که آیا هر لیستی که لود می‌شود واقعا 24 تا آگهی (tokens) دارد؟ طبق عکس زیر معلوم می‌شود که حدود 3020 تا از لیست‌ها اصلا ۲۴ تا آگهی در خود ندارند، که برخلاف اطلاعاتی است که در مقدمه‌ی دستورِ پروژه آمده.

```
In [118]:
#Just to make sure there are 24 ads in each loaded list:

flag = 0
lst_index_null_tokens = data[data['tokens'].isnull() == True]['tokens'].index.to_list()
lst_len = []

for i in range(0, len(data['tokens'])):
    if i not in lst_index_null_tokens and len(data['tokens'][i].split(',')) != 24:
        flag += 1
        lst_len.append([i, len(data['tokens'][i].split(','))])

print(flag)
```

```
3020
```

```
In [119]:
lst_len
```

```
Out[119]: [[6, 10],
 [32, 8],
 [33, 7],
 [57, 7],
 [67, 21],
 [105, 11],
 [117, 7],
 [118, 7],
 [124, 4],
 [131, 3],
 [136, 4],
 [141, 20],
 [152, 23],
 [157, 8],
 [158, 8],
 [163, 1],
```

Figure 9, Not all of the loaded lists have 24 tokens!

این ممکن است به خاطر اختلال در اینترنت یوزر هم بوده باشد، یا اینکه کوئری ها به درستی اجرا نشده اند. ولی مهمترین دلیل به نظر من این است که کوئری که یوزر استفاده کرده اصلا 24 تا دیتا در خود نداشته که بخواهد لود شود، بلکه تعداد کالاهایی که آن یوزر با آن مشخصات مد نظر خود زده است،کمتر از 24 تا بوده است. این مورد، در سوال 3 قسمت الف حائز اهمیت است! زیرا نمی‌توانیم برای به دست آوردن تعداد کل آگهی‌های لود شده، صرفا تعداد لیست‌ها را در 24 ضرب کنیم! چون هر لیستی می‌تواند ماکزیمم و نه لزوما 24 تا آگهی داشته باشد! پس باید تعداد اگهی‌های هر لیست مربوطه را جمع کنیم. لیست all_tokens_lst به ترتیب شامل تعداد توکن‌های لود شده در هر لیست است.

```
#Just to make sure there are 24 ads in each loaded list:

flag = 0
lst_index_null_tokens = data[data['tokens'].isnull() == True]['tokens'].index.to_list()
# lst_tokens_not_full_24 = []
all_tokens_lst = []

for i in range(0, len(data['tokens'])):
#     if i not in lst_index_null_tokens and len(data['tokens'][i].split(',')) != 24:
#         flag += 1
#         lst_tokens_not_full_24.append([i, len(data['tokens'][i].split(','))])
    if i not in lst_index_null_tokens:
        all_tokens_lst.append(len(data['tokens'][i].split(',')))

# print(flag)
# print('*')
# print(lst_tokens_not_full_24)
```

Figure 10, Calculating the number of ads (tokens) in each list of the query

- نکته دیگر این است که در ستون post_index_in_post_list، تایپِ داده‌ها بر اساس float64 ثبت شده است در حالی که این داده‌ها نشان‌دهنده‌ی رتبه آگهی در لیست اند و اعداد صحیحی هستند. پس شاید بهتر باشد که داده‌های این ستون را به int تبدیل کرد. این کار در سوال 3، برای قسمت دوم انجام شده است.

```
In [148]: data_click = data[data['action'] == 'click_post'].sort_values(by = 'source_event_id')

In [149]: type(data_click['post_index_in_post_list'].iloc[0])
Out[149]: numpy.float64

In [150]: data_click['post_index_in_post_list'] = data_click['post_index_in_post_list'].astype('int')

In [151]: type(data_click['post_index_in_post_list'].iloc[0])
Out[151]: numpy.int32
```

Figure 11, Converting data type "post_index_in_post_list" from float to integer

سوال دوم)

الف) محاسبه‌ی dark query percent:

برای محاسبه این متریک، باید ابتدا داده‌ها را بر اساس ستون source_event_id دسته‌بندی کنیم و سپس آن‌هایی را که کمتر از ده نتیجه نشان می‌دهند ذخیره کنیم. سپس نسبت به تعداد کل داده‌ها، درصد می‌گیریم. خروجی زیر نشان می‌دهد که برای حدود 9.95% کوئری‌ها، کمتر از ده نتیجه نمایش ثبت شده است.

```
#Metric 1: dark query percent

data_groupby = data.groupby(by = 'source_event_id').count() < 10
less_than_ten = data_groupby[data_groupby['action'] == True]['action'].count()
dark_percent = less_than_ten / len_data * 100

print('The Dark Query Percent Metric is {0:.2f}%.' .format(dark_percent)) #Displaying with only two deciaml points
```

```
The Dark Query Percent Metric is 9.95%.
```

Figure 12, The dark query percent metric

ب) محاسبه‌ی query bounce rate:

برای محاسبه این متریک، باید ابتدا تعداد کوئری‌های load شده را به دست بیاوریم. این کار با متد count() که تعداد سطرهای غیر NaN را می‌شمارد امکان‌پذیر است. همان طور که در خروجی قابل مشاهده است، 31.77% کوئری‌ها مربوط به حالتی است که کاربر روی هیچ کدام از نتایج کلیک نکرده است.

```
#Metric 2: query bounce rate

load = data['post_page_offset'].count() #Or we could use variable "num_loadings". We could also use "data['tokens'].count()"
bounce_rate = (load/len_data) * 100

print('The Query Bounce Rate Metric is {0:.2f}%.' .format(bounce_rate)) #Displaying with only two deciaml points
```

```
The Query Bounce Rate Metric is 31.77%.
```

Figure 13, The Query Bounce Rate Metric

سوال سوم)

الف) محاسبه‌ی درصدآگهی‌های کلیک شده نسبت به آگهی‌های لود شده

ابتدا باید تعداد آگهی‌های کلیک شده را به دست بیاوریم. سپس در لیستی که شامل تعداد آگهی‌هایِ در هر لیستِ لود شده است (در قسمت سوال ۱ توضیح داده شد)، تعداد آگهی‌ها را جمع می‌کنیم. باید دقت شود که همان‌طور که در سوال ۱ توضیح داده شد، درست نیست که تعداد لیست‌ها را در 24 ضرب کنیم به گمان اینکه همه‌ی لیست‌های لود شده دارای 24 آگهی هستند. طبق خروجی زیر، درصدآگهی‌های کلیک شده نسبت به آگهی‌های لود شده برابر با 9.43% است و این یعنی بیشتر یوزرها صرفا scroll می‌کنند و یا اینکه آگهی‌ها برایشان به اندازه کافی جذاب نیست.

```
#The number of loads abd clicks were claculated in the previos section. So we use them here:

click = data['post_index_in_post_list'].count() #We should make sure we are NOT using "post_token" since it has one less entr
click_load_percent = click / (np.array(all_tokens_lst).sum()) * 100
print('The percentage of the clicked ads to the loaded ads is {0:.2f}%.' .format(click_load_percent)) #Displaying with only t
```

```
The percentage of the clicked ads to the loaded ads is 9.43%.
```

Figure 14, The percentage of the clicked adds to the total loaded ads

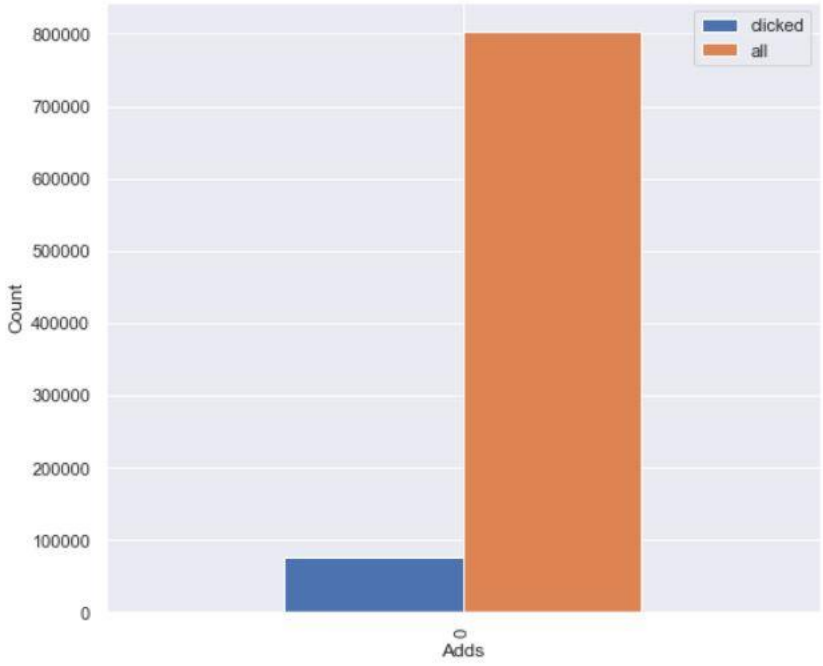

Figure 15, Bar Plot of the clicked adds (blue) vs all of the loaded adds (orange)

ب) رتبه اولین کلیک کاربر (برای کوئری‌هایی که حداقل یک کلیک داشته‌اند)

برای این قسمت، ابتدا دیتاهای مربوط به کلیک کردن بر اساس کوئری (source_event_id)، مرتب (سورت) شد. سپس از آن‌جایی که در سوال ۱ هم توضیح داده شد، رتبه آگهی‌ها به صورت اعشاری با یک رقم اعشار 0 بود که آن را به عدد صحیح تبدیل کردم. سپس، دیتاها براساس زمان (یعنی ستون) و کوئری (یعنی ستون) مرتب شد (می‌توانستیم براسای کاربر یعنی ستون devide_id هم مرتب‌سازی کنیم). حال چون اولین ردیف برای هر کوئری در این حالتِ مرتب شده، نشان‌دهنده‌ی اولین کلیک روی آن کوئری است، پس به ازای هر کوئری متمایز، اولین ردیف مربوط به آن کوئری را ذخیره می‌کنیم.

	action	created_at	source_event_id	device_id	post_page_offset	tokens	post_index_in_post_list	post_token
41981	click_post	1609545206093	00142c59-745c-4004-a955-698ddcf1faa6	Yt1KNNvXQAqN7wUyh8F48Q	NaN	NaN	7	wXuDbcdY
13136	click_post	1609546723661	0016f59f-9fb3-4ab5-ae78-9783314b81fc	AIIjxxZPQKaPTnEX8aemNA	NaN	NaN	2	wXvfYPCE
35290	click_post	1609544967604	0017b9ef-5903-40f9-a219-85728eb78436	v_pt772LRB2HRoGGCr1zTA	NaN	NaN	5	wXvrEhKR
60420	click_post	1609543868493	0017c208-e016-4231-a022-598ba020f1ff	3o5XzXgSQ-i7_LQpts8jxQ	NaN	NaN	2	wXj3iwnq
49915	click_post	1609545218693	00240c5c-9d06-4fcd-93f5-93b842fdab63	NWJ25xuOSuG4vbWkYB534Q	NaN	NaN	1	wXR7b2eM
...	...	...	...	...	...	...	...	...
43750	click_post	1609545448055	ffdd48fc-4c05-4c75-bf8c-5cd42729d2b8	CKM0dYauTMi2XsRF8riGPQ	NaN	NaN	5	wXvXB3tl

Figure 16, for each of the distinctive query ids

همچنین من میانگین رتبه‌های اولین کلیک کاربران برای همه‌ی کوئری‌های متمایز را نیز محاسبه کردم که به صورت زیر است:

```
data_first_clicks['post_index_in_post_list'].mean()
```

```
26.658595194085027
```

Figure 17, The mean of all the first clicked adds for all the distinctive query ids

پ) میانگین فاصله بین رتبه کلیک‌های یک کاربر (مثلا اگر کاربر از بین نتایج کوئری روی آگهی‌های دهم و شانزدهم کلیک کند، این متریک برابر (10+6)/2 یعنی 8 خواهد بود)

در اینجا از دیتای مربوط به کلیک شدن که براساس زمان در قسمت‌های قبل سورت شد، استفاده می‌شود. سپس نسبت به کوئری‌ها (یعنی ستون source_event_id) گروه‌بندی می‌کنیم و میانگین فاصله بین رتبه کلیک‌های یک کاربر را محاسبه می‌کنیم.

```
#Group by the query IDs ("source_event_id"): -> query related

data_click.groupby(by = 'source_event_id')['post_index_in_post_list'].mean()
```

```
source_event_id
00142c59-745c-4004-a955-698ddcf1faa6     71.750000
0016f59f-9fb3-4ab5-ae78-9783314b81fc      9.700000
0017b9ef-5903-40f9-a219-85728eb78436     10.000000
0017c208-e016-4231-a022-598ba020f1ff     20.750000
00240c5c-9d06-4fcd-93f5-93b842fdab63      1.000000
                                           ...
ffdd48fc-4c05-4c75-bf8c-5cd42729d2b8     12.800000
ffe3a1f8-c363-4fbb-af5f-0d244b21aea4     52.714286
ffec4e11-c0a1-4fa4-8613-0979d6f46918    318.800000
ffef31ea-cb4e-4d00-98ca-118df330cdca     34.000000
ffef6048-9db0-4244-9750-3b530552f0ec    376.395349
Name: post_index_in_post_list, Length: 10820, dtype: float64
```

Figure 18, The mean of the "distances" between the clicked adds for distinctive queries for the users, grouped by the query ids

میانگین خروجی به دست آمده برای همه‌ی کوئری‌ها برابر است با:

```
data_click.groupby(by = 'source_event_id')['post_index_in_post_list'].mean().mean()
```

```
54.96911150862862
```

Figure 19, The mean of all the "mean-distances" for all the queries in the dataset, based on query ids

همچنین اگر به جای کوئری‌ها براساس یوزرها (یعنی device_id) گروه‌بندی کنیم، خواهیم داشت:

```
#Or group by the device IDs ("device_id"): -> user related

data_click.groupby(by = 'device_id')['post_index_in_post_list'].mean()
```

```
device_id
-0CO6Hc_QiqH4T6IRnAkMw      1.000000
-0IGwculQbu66z8q_Vm9MA    262.960526
-1wIhljjS_yhOisAOhIdNA     17.500000
-2HNBT8SRiC3klN90eGvNQ     21.312500
-2Hh978WRIuDcnICSpbzjg     18.000000
                               ...
zsozTupGRlOefdBRLVoNKQ     59.512821
ztcsKyk4Tw-nw7Q8PIERsQ      1.000000
zwF_8nudT1aCBJpDmGtCWw      3.500000
zyiz3SbnRcSgRb89NgiQ2A     72.000000
zzfgAZsyTjOltwjV6AWcjg     42.448276
Name: post_index_in_post_list, Length: 4550, dtype: float64
```

Figure 20, The mean of the "distances" between the clicked adds for distinctive queries for the users, grouped by the device ids

میانگین خروجی به دست آمده برای همه‌ی یوزرها برابر است با:

```
data_click.groupby(by = 'device_id')['post_index_in_post_list'].mean().mean()
```

```
79.77753994597528
```

Figure 21, The mean of all the "mean-distances" for all the queries in the dataset, based on device ids

ت) اینکه آیا روی یکی از 3 نتیجه اول کوئری کلیک شده یا نه.

ابتدا مشابه قسمت قبل، از دیتای مربوط به کل کوئری‌های کلیک شده که براساس آی‌دی کوئری‌ها گروه‌بندی شده اند استفاده می‌کنیم و چک می‌کنیم که روی یکی از سه نتیجه اول کوئری کلیک شده یا نه، یعنی مقدار داده‌ی موجود در ستون post_index_in_post_list نباید بیشتر از ۳ باشد.

```
data_click[data_click['post_index_in_post_list'] <= 3]
```

	action	created_at	source_event_id	device_id	post_page_offset	tokens	post_index_in_post_list	post_token
71120	click_post	1609512693708	1b427086-fac9-4bdc-a87f-d980ca139bd5	f-NfqjIwSHyIPtw1Z5YR6A	NaN	NaN	2	wXtnb0R5
96683	click_post	1609539419836	5663fc34-6876-4a46-a7bb-2f2be1bc40a3	JcYXHfWSSDSOrz6JSf6xYQ	NaN	NaN	2	wXvDrSIa
98351	click_post	1609542798049	0c3b2258-a8c6-4408-80f7-e4b992f706b0	YZLUosstTjKLkAS1Nj0DXw	NaN	NaN	2	wXvTWcMf
76828	click_post	1609543073167	b2b6e5c6-422a-423b-82b2-97f66f1452e2	ovByBatATHivJb9Uh8LH6A	NaN	NaN	3	wXvXKiuD
16683	click_post	1609543625330	5464ee74-0608-4451-ae8c-fb5e85466116	2y5EIUTBQGuznrh3b4MrVQ	NaN	NaN	1	wXvb5jwt
...	...	...	...	...	...	...	...	...
79364	click_post	1609587946849	c4104175-7d63-49a1-b90a-b2ed571f113d	_2pankxtTMOiYWvZ4SPmvg	NaN	NaN	1	wXvTZvb1
71295	click_post	1609587973806	c4104175-7d63-49a1-b90a-b2ed571f113d	_2pankxtTMOiYWvZ4SPmvg	NaN	NaN	2	wXvP2gww
31706	click_post	1609587979760	c4104175-7d63-49a1-b90a-b2ed571f113d	_2pankxtTMOiYWvZ4SPmvg	NaN	NaN	3	wXsHUV4_
26557	click_post	1609632927251	65602350-d258-47f6-b854-cedfc122aa18	Q_Ygkm1cSDaU7_NsoUcA9w	NaN	NaN	2	wXvT7snI
9796	click_post	1609633388621	86dd16f9-8356-4a5b-8ad2-be903db7b60a	Q_Ygkm1cSDaU7_NsoUcA9w	NaN	NaN	1	wXvn6Zo6

7496 rows × 8 columns

Figure22, The queries where the user clicked one of the first three results

تعداد کوئری‌هایی که روی یکی از سه نتیجه اولشان کلیک شده در این حالت برابر است با:

```
data_click[data_click['post_index_in_post_list'] <= 3]['post_index_in_post_list'].count()
```

```
7496
```

Figure 23, The number of queries where the users clicked on one of the first three results

همچنین، درصد تعداد این کوئری‌ها نسبت به کل کوئری‌های مربوط به کلیک کردن برابر است با:

```
#Percentage of clicking on the first three results VS all of the clicked queries:

num_three = data_click[data_click['post_index_in_post_list'] <= 3]['post_index_in_post_list'].count()
three_clicks_to_all = num_three / num_clicks * 100

print(three_clicks_to_all)
```

```
9.889703942160537
```

Figure 24, The number of queries where the users clicked on one of the first three results

بررسی متریک‌های معرفی شده به ترتیب:

الف)

درصد آگهی‌های کلیک شده نسبت به آگهی‌های لود شده برابر با 9.43% است که مقدار کمی است. این می‌تواند نشانگر این باشد که آگهی‌ها واقعا مطابق خواسته‌ی کاربران نبوده اند (-). همچنین می‌تواند نشانگر این باشد که تعداد کاربران سایت زیاد است و تعداد آگهی‌های زیادی را لود کرده اند، درنتیجه درصد کلیک به لود پایین آمده است (+). و یا می‌تواند نشانگر این باشد که آگهی‌ها دارای تنوع بالایی اند و کاربران ترجیح دادند لیست آگهی‌های بیشتری را لود کنند تا بهترین انتخاب خود را کلیک کنند (+). برای این که مطمئن شویم این حالت برقرار است یا نه، می‌توانیم از متریک قسمت (ب) استفاده کنیم تا ببینیم به طور میانگین، کاربران روی چندمین آگهی برای اولین بار کلیک می‌کنند که به طور میانگین حدودا برابر با 26 است. یعنی حدودا لیست اول و کمی از لیست دوم. به نظر می‌آید آگهی‌ها توانسته اند کاربران را قانع کنند که به جای سریع کلیک کردن، آگهی‌های بیشتری را لود کنند.

ب)

همان‌طور که در قسمت (الف) اشاره شد، رتبه اولین کلیک کاربر به طور میانگین حدود 26 است. یعنی با توجه به اینکه به طور میانگین هم حدود 24 آگهی در هر لیست وجود دارد، هر کاربر به طور میانگین حداقل یک لیست از آگهی‌ها را تا آخر چک می‌کند.

این یا می‌تواند به دلیل تنوع خوب و بالای آگهی‌ها باشد که یوزر را ترغیب می‌کند که به لود کردن آگهی‌های بیشتر ادامه دهد (+)، و یا اینکه ممکن است نشانه این باشد که نتایج اولیه‌ی نشان داده شده به یوزر به اندازه کافی مطابق خواسته او نبوده و او باید آگهی‌های بیشتری را لود می‌کرده تا کالای مورد نظر خود را پیدا کند(-). برای اینکه بفهمیم کدام مورد بوده، متریک قسمت (پ) می‌تواند کمک‌کننده باشد، زیرا نشان می‌دهد کلیک‌های بعدی کاربر به ظور میانگین چه فاصله‌ای با هم داشتند. البته بدی این روش این است که اگر یک داده پرت (یک کلیک که با کلیک قبلی خود فاصله زیادی دارد) وجود داشه باشد، می‌تواند روی مقدار این میانگین کلی هم تاثیر بگذارد و به اشتباه، میانگین کلی را بالا ببرد. در این مواقع استفاده از median گزینه بهتری است.

پ)

میانگین فاصله بین رتبه کلیک‌های یک کاربر نیز می‌تواند نشان دهد که کاربر تا چند تا لیستِ بعدی پس از کلیکِ اول را هم لود کرده است و مطالب چه قدر برای او جذاب بوده اند. اگر نسبت به آی‌دی کوئری‌ها بخواهیم این مقدار را به دست آوریم، میانگین فاصله حدودا برابر 60 است. که یعنی حدود $60 \div 24 = 2.5$ تا لیست را بعد از حداقل اولین کلیک، لود کرده است. پس یعنی یوزر به گشتن ادامه داده و حداقل ناامید یا منصرف نشده، اما از طرفی همان‌طور که توضیح داده شد، اولین آگهی‌های نشان داده شده هم او را سریعا جذب نکرده اند، زیرا در قسمت (ب) نشان داده شد که اولین کلیک به طور میانگین، 26 امین آگهی بوده، نه زودتر. برای اینکه این مورد را واضح تر ببینیم، به متریک بعدی در قسمت (ت) نیاز داریم تا چک کنیم حدودا چند بار روی یکی از سه نتیجه اول کوئری کلیک شده.

همچنین، میانگین فاصله بین رتبه کلیک‌های یک کاربر را نسبت به هر یوزر (device_id) هم حساب کردم که حدودا برابر با 80 شد که منطقی است که با قسمت قبل تفاوت چشمگیری نداشته باشد از این جهت که سلیقه یک کاربر در طول کوئری‌های مختلفی که می‌زند، تغییر خاصی نمی‌کند.

ت)

در این قسمت، تعداد کوئری‌هایی که روی یکی از سه نتیجه اولشان کلیک شده را چک می‌کنیم. همان‌طور که در سوال 3 بیان شد، به تعداد 7496 تا کوئری با این ویژگی وجود دارد که تنها حدود 9.89% از تعداد کل کوئری‌ها را تشکیل می‌دهند که صحتی است بر آنچه در قسمت (پ) و (ب) بیان شد. یعنی، کاربران سریعا روی لینک‌های اول کلیک نمی‌کنند بلکه ترجیح میدهند به طور میانگین روی 26 امین آگهی برای اولین بار کلیک کرده و سپس با فاصله میانگین حدود 60 تا، روی آگهی‌های بعدی کلیک کنند.

به طور یک جمع‌بندی کلی، هر چهار تای این متریک‌ها کمک می‌کنند تا بفهمیم آگهی‌های نشان داده شده، به چه میزان مطابق خواسته کاربران بوده است. اما اگر بخواهم خلاصه‌تر انتخاب کنم، متریک‌های قسمت (ب) و (پ) واضح‌تر این مورد را نشان می‌دهند. میانگین و توضیحات مربوطه شان هم در بالا بیان شد.

سوال چهارم)

!!!