به نام او

تسک ورودی دورهٔ تابستانی
شرکت دیوار

محمدحسین کلهر

سوال ۱

از خطاهای موجود در داده می توان به موارد زیر اشاره کرد:

- تعدادی کوئری در داده وجود دارد که فقط اکشن `click_post` آن‌ها ثبت شده است و اکشن `load_post_page` برای آن‌ها موجود نیست.

سوال ۲

ابتدا کتابخانه‌های لازم را import می کنیم:

[1]:
```
import pandas as pd
import numpy as np
```

و سپس داده را می‌خوانیم:

[2]:
```
data = pd.read_excel('Documents/DivarTask/Summer Camp Task Data.xlsx')
```

محاسبه‌ی dark query percent

می‌خواهیم تعداد آگهی‌های لود شده را حساب کنیم. ستون `tokens` از نوع رشته (string) است. بنابراین باید ابتدا آن را به لیستی از رشته‌ها تبدیل کنیم و سپس طول این لیست را به عنوان تعداد آگهی‌های لود شده در نظر بگیریم. تابع `tok` این کار را انجام می‌دهد. البته طبق مشاهده‌ای که داشتیم، شناسه‌ی همه‌ی تبلیغ‌ها یک رشته به طول ۸ است و می‌توانستیم بدون تعریف تابع و با توجه به این موضوع تعداد آگهی‌های لود شده را بدست بیاوریم اما برای اطمینان از تابع زیر استفاده می‌کنیم.

[3]:
```
def tok(str):
    str = str.replace('[','')
    str = str.replace(']','')
    tokens = str.split(',')
    return tokens
```

حالا با توجه به این که هر بار لود، شامل ۲۴ آگهی است، پس کوئری‌هایی که برای آن‌ها کمتر از ۱۰ نتیجه نمایش دادیم مقدار `post_page_offset` آن‌ها برابر صفر است و طول لیست شناسه‌های آگهی‌های لود شده‌ی آن‌ها نیز کمتر از ۱۰ است. پس ابتدا این سطرها را فیلتر می‌کنیم و تابع `tok` را بر ستون `tokens` آن‌ها اعمال می‌کنیم.

[4]:
```
filtered = data.loc[data['post_page_offset'] == 0].reset_index()
numb_ads = filtered['tokens'].apply(tok).apply(len)
```

حالا سطرهایی را جدا می‌کنیم که تعداد آگهی‌های آن‌ها کمتر از ۱۰ است و متغیر `numb_dark` را برابر تعداد این سطرها قرار می‌دهیم.

[5]:
```
numb_dark = len(numb_ads[numb_ads < 10])
numb_dark
```

[5]:
```
1190
```

حالا تعداد کل کوئری‌ها را بدست می‌آوریم.

[6]:
```
numb_query = data.source_event_id.unique().size
numb_query
```

[6]:
```
14296
```

و درنهایت با تقسیم تعداد کوئری‌های با تعداد نتایج کمتر از ۱۰ بر کل تعداد کوئری‌ها dark query percent را بدست می‌آوریم:

[7]:
```
dark_query_percent = numb_dark/numb_query
dark_query_percent
```

[7]:
```
0.08324006715165082
```

همانطور که در خروجی بالا قابل مشاهده است این مقدار برابر با ۸٫۳ درصد است.

محاسبه‌ی query bounce rate

می‌خواهیم کوئری‌هایی را پیدا کنیم که فقط اکشن load_post_page را دارند. این یعنی که کاربر روی هیچ کدام از نتایج این کوئری‌ها کلیک نکرده است. مجموعه‌ی load_post_query کوئری‌هایی است که برای آن‌ها آگهی لود شده و مجموعه‌ی click_post_query کوئری‌هایی است که کاربر روی بعضی از نتایج آن کلیک کرده است.

[8]:
```
load_post_query = set(data[data['action'] ==␣
 ↪'load_post_page']['source_event_id'].unique())
click_post_query = set(data[data['action'] ==␣
 ↪'click_post']['source_event_id'].unique())
```

حالا برای بدست آوردن کوئری‌هایی که هیچ کلیکی روی نتایج آن‌ها نشده است کافی‌ست تفاضل این دو مجموعه را در نظر بدست بیاوریم. لیست unclicked کوئری‌هایی‌ست که در load_post_query هستند ولی در click_post_query نیستند. پس این کوئری‌های کلیکی نداشته‌اند. طول این لیست را در numb_unclicked ذخیره می‌کنیم.

[9]:
```
unclicked = list(load_post_query-click_post_query)
numb_unclicked = len(unclicked)
numb_unclicked
```

[9]:
```
3476
```

حالا numb_unclicked را بر تعداد کل کوئری‌ها تقسیم می‌کنیم تا نرخ کوئری‌های بدون کلیک، query bounce rate، را بدست بیاوریم.

[10]:
```
query_bounce_rate = numb_unclicked/numb_query
query_bounce_rate
```

[10]:
```
0.24314493564633463
```

همانطور که در خروجی بالا قابل مشاهده است این مقدار برابر با ۲۴٫۳ درصد است.

سوال ۳

به نظر می‌رسد معیار اول، معیار به نسبت بهتری باشد. البته نگارنده دفاع محکمی برای این ادعا ندارد. ولی به طور شهودی برای اغلب کاربران کلیک کردن روی آگهی، یعنی آگهی کلیک شده برای آن‌ها جلب توجه کرده و احتمالا به خواسته‌ی آن‌ها نزدیک بوده است. یک نکته‌ی دیگر این که با توجه به این که دیوار آگهی‌ها را بر اساس زمان انتشار نمایش می‌دهد بنابراین نباید کلیک روی تبلیغ‌های از لحاظ مکانی پایین‌تر (رتبه‌ی بیشتر) آن‌چنان بد تلقی شود. تبلیغ‌های ابتدایی هم به طور طبیعی بیشتر کلیک می‌خورند. زیرا کاربر به طور پیش‌فرض احساس می‌کند تبلیغ‌های ابتدایی مرتبط‌تر با خواسته‌ی او هستند. پس نسبت کلیک‌ها به کل آگهی‌ها شاید معیار بهتری برای موفقیت در نمایش آگهی‌های مرتبط باشد. به هر حال شاید این معیارها

در کنار هم توضیح‌دهندگی بیشتری داشته باشند. در ادامه هر چهار معیار محاسبه شده‌اند. توضحیح کد مربوط به متریک اول در ادامه آمده است.

ابتدا برای این که داده‌ی اصلی را تغییری ندهیم یک کپی از آن را در copy_data ذخیره می‌کنیم.

[11]:
```
copy_data = data.copy()
```

یک ستون جدید به داده اضافه می‌کنیم. ستون numb_clicks برای سطرهایی که اکشن آن‌ها load_post_page است برابر با صفر و برای سطرهایی که اکشن آن‌ها click_post است برابر با یک است. در این صورت جمع خانه‌های این ستون برای هر کوئری نشان می‌دهد که هر کوئری در مجموع چند کلیک داشته است.

[12]:
```
copy_data.loc[copy_data['action']=='click_post', 'numb_clicks'] = 1
copy_data.loc[copy_data['action']=='load_post_page', 'numb_clicks'] = 0
```

ستون numb_ads هم به داده اضافه می‌کنیم که تعداد تبلیغ هر اکشن را نشان دهد. طبعیتا این مقدار برای سطرهای با اکشن click_post برابر صفر و برای سطرهای با اکشن load_post_page برابر طول لیست آگهی‌هاست. (علت تعریف کردن تابع tok و کاربرد آن در سوال ۲، بلوک کد ۳، آمده است). مجموع مقادیر این ستون برای هر کوئری نشان می‌دهد که هر کوئری در مجموع حاوی چند آگهی بوده است.

[13]:
```
copy_data['numb_ads'] = copy_data.
 ↪loc[copy_data['action']=='load_post_page']['tokens'].apply(tok).
 ↪apply(len)
copy_data.loc[copy_data['action']=='click_post', 'numb_ads'] = 0
```

حالا با توجه به توضیحات بالا مجموع مقادیر این دو ستون را برای هر کوئری حساب می‌کنیم و با تقسیم تعداد کلیک‌ها به تعداد آگهی‌های لود شده نرخ کلیک را برای هر کوئری حساب می‌کنیم. ستون clikcs_rate نرخ کلیک برای هر کوئری است.

[14]:
```
query_data = copy_data.groupby('source_event_id',␣
 ↪as_index=False)[['numb_ads', 'numb_clicks']].agg(np.sum)
query_data['clicks_rate'] = query_data['numb_clicks']/
 ↪query_data['numb_ads']
query_data.loc[query_data['numb_ads']==0, 'clicks_rate'] = 0
query_data.aggregate(np.mean)
```

[14]:
```
numb_ads       56.213556
numb_clicks     5.301903
clicks_rate     0.062037
dtype: float64
```

با توجه به خروجی بالا میانگین نرخ کلیک برای همه‌ی کوئری‌ها برابر با ۶٫۲ درصد است. البته با توجه به این که تعداد کوئری وجود دارد که فقط اکشن کلیک برای آن‌ها ثبت شده است و اکشن لود آن‌ها در داده نیست ممکن است این مقدار کمی خطا داشته باشد.

در ادامه‌ی این سوال باقی معیارها نیز محاسبه شده‌اند که با توجه به این که در صورت سوال خواسته نشده‌اند از توضیح آن‌ها خودداری می‌کنیم.

[15]:
```
copy_data['created_at'] = copy_data['created_at'].apply(pd.Timestamp,␣
↪unit='ms')
```

محاسبه‌ی رتبه‌ی اولین کلیک کاربر:

[16]:
```
temp_data = copy_data.loc[copy_data['action']=='click_post'].
↪groupby('source_event_id', as_index=False)['created_at'].idxmin()
click = copy_data.loc[temp_data['created_at'],:
↪][['source_event_id','post_index_in_post_list']].reset_index()
del click['index']
click = click.rename(columns={'post_index_in_post_list':
↪'first_click_index'})
```

[17]:
```
query_data = pd.merge(left = query_data, right = click, how='left')
```

محاسبه‌ی میانگین فاصله‌ی بین رتبه‌ی کلیک‌های کاربر: چیزی که از توضیحات صورت سوال متوجه شدم این معیار معادلا برابر است با رتبه‌ی آخرین کلیک تقسیم بر تعداد کلیک‌ها.

[18]:
```
temp_data1 = copy_data[copy_data['action']=='click_post'].
↪groupby('source_event_id', as_index=False)['post_index_in_post_list'].
↪agg('count')
temp_data1 = temp_data1.rename(columns={'post_index_in_post_list':␣
↪'count'})
temp_data2 = copy_data[copy_data['action']=='click_post'].
↪groupby('source_event_id', as_index=False)['post_index_in_post_list'].
↪agg('max')
temp_data = pd.merge(left=temp_data1, right=temp_data2, how="left")
temp_data['avg_index_dist'] = temp_data['post_index_in_post_list']/
↪temp_data['count']
del temp_data['count']
del temp_data['post_index_in_post_list']
```

[19]:
```
query_data = pd.merge(left = query_data, right = temp_data, how='left')
```

بررسی این که آیا روی یکی از ۳ نتیجه اول کوئری کلیک شده یا نه:

[20]:
```
temp_data = data.groupby('source_event_id',␣
 ↪as_index=False)['post_index_in_post_list'].agg(np.min)
condition = temp_data['post_index_in_post_list']<4
temp_data['first_three_ads'] = np.where(condition,1,0)
del temp_data['post_index_in_post_list']
```

[21]:
```
query_data = pd.merge(left = query_data, right = temp_data, how='left')
```

[22]:
```
query_data.aggregate(np.mean)
```

[22]:
```
numb_ads             56.213556
numb_clicks           5.301903
clicks_rate           0.062037
first_click_index    26.658595
avg_index_dist       15.366037
first_three_ads       0.354085
dtype: float64
```

در خروجی بالا نیز میانگین هر کدام از معیارها را برای همه‌ی کوئری‌ها قابل مشاهده است.

سوال ۴

توزیع برنولی با یک پارامتر p مشخص می‌شود. این پارامتر تعیین می‌کند که در آزمایش دو حالته‌ی مورد نظر احتمال وقوع هر کدام از دو حالت چقدر است. مدل‌سازی ما نیز برای توضیح رفتار کاربران می‌تواند به همین شکل باشد. یعنی می‌خواهیم پیش‌بینی کنیم که احتمال کلیک شدن هر کدام از نتایج کوئری‌ها توسط کاربر چقدر است. پس کلیک شدن/نشدن حالت‌های آزمایش ما هستند. حالا باید پارامتر p را مشخص کنیم. این پارامتر به این معناست که هر تبلیغ یک کوئری توسط کاربر به احتمال p کلیک خواهد شد و به احتمال $p - ۱$ این اتفاق نخواهد افتاد. برای مشخص کردن پارامتر p می‌توانیم از متریک محاسبه شده در سوال قبل استفاده کنیم. در قسمت قبل محاسبه کردیم که میانگین نرخ کلیک روی نتایج کوئری‌ها برابر با ۶٫۲ درصد است. بنابراین یک انتخاب مناسب برای p همین مقدار ۰٫۰۶۲ است. متاسفانه ارتباط متریک‌های گفته شده در سوال قبل برای نگارنده روشن نیست. همچنین جهت شفافیت، برای پاسخ به این سوال کمی جستجو کردم اما به نتیجه‌ی قابل ملاحظه‌ای نرسیدم اما کمی از این لینک کمک گرفتم.