ارائه تسک ورودی سامر کمپ دیوار

ارائه دهنده : محمد عرب انواری

تعریف مسئله (با چی سر و کار دارم)

من دیتای رفتار کاربران اپلیکیشن دیوار در یک روز خاص رو در اختیار دارم که شامل ۱۱۱۰۸۳ سطر و ۸ ستون میشه.
دو سطر انتخابی از این دیتا رو توی جدول زیر میبینیم.

action	created_at	source_event_id	device_id	post_page_offset	tokens	post_index_in_post_list	post_token
load_post_page	1609545001150	1575558c-a702-46ef-8e18-bc5cef761473	Ed9EADRZRXCHcrcEP_Hnkg	26.0	[wXvP3enu, wXvHXzUS, wXvPHXVe, wXvPHRs3, wXvH3...	NaN	NaN
click_post	1609544675058	8ffcb67f-0c0b-4d98-8cc3-bf251920692c	8NjLp6swSX2beaez5ogFrg	NaN	NaN	71.0	wXkzIqTL

چه مسئله هایی تعریف شدن؟

توی این تسک چهار تا مسئله داده شده که به شرح زیر هستن :

۱. **پیدا کردن خطا های احتمالی در دیتا:**
که این کار رو با تست سناریو های مختلف بر اساس الگوهایی که توی داده پیدا میشه انجام دادم.
برای مثال بر اساس صورت مسئله تمام داده ها مربوط به یک روز هستند، ایا اینطوریه؟ یا device_id ها در اکثر ستون ها یک رشته به طول ۲۲ هستند ایا غیر از این وجود داره؟ اگر بله، آیا خطا حساب میشه؟
۲. **محاسبه ی دو متریک خواسته شده:**
این دو متریک رو هم با متد های pandas به خوبی تونستم محاسبه کنم و شرح بیشتر رو در ادامه میدم.
۳. **انتخاب یک متریک از بین چهار متریک داده شده برای تشخیص مناسب بودن یک کوئری:**
این متریک رو با توجه به این که اگهی های دیوار ،برخلاف سرویس هایی مثل گوگل، به ترتیب زمان نشون داده میشن، انتخاب کردم. و در ضمن یک متریک هم که به نظرم بهتر میتونست کمکمون کنه پیشنهاد کردم.
۴. **مطالعه روی توزیع برنولی و تشخیص اینکه ایا بعد از مدلسازی کلیک کاربرها با این توزیع، این متریک ها از روی هم بدست میان یا نه:**
توزیع برنولی یک حالت خاص از توزیع دوجمله ای هست با توجه به این حقیقت، داده ها رو با توزیع دو جمله ای مدل کردم، و با توجه به درکی که از این توزیع داشتم به سوال مورد نظر هم جواب دادم.

در ادامه به ترتیب کار هایی که انجام دادم رو به اختصار توضیح میدم و به سوال هایی که پرسیده شده جواب میدم.

آماده سازی دیتا:

حل مسئله زمان:

زمان داده شده در ستون `created_at` به فرمت timestamp هست. و خب برای اینکه داده های این ستون قابل درک تر باشن اون ها رو با این دو خط کد به `datetime` تبدیل کردم.
خط دوم برای همگام سازی زمان با timezone ایران اضافه شده.

```
data['created_at']=pd.to_datetime(data['created_at'],unit='ms')
data['created_at']=data['created_at']+pd.Timedelta('03:30:00')
```

چطور به مقادیر ستون tokens دسترسی بهتری داشته باشیم؟

مشخصه که داده های ستون `tokens` اول به صورت لیست و در یک دیتا فریم بودن اما بعد از ذخیره شدن و لود شدن به صورت یک رشته دراومدن و این فرمتی نیست که من بتونم به سادگی به اجزاش دسترسی داشته باشم پس با تکه کد زیر یک ستون جدید به اسم `tokens_list` ساختم و رشته های تبدیل به لیست شده‌ی ستون `tokens` رو اونجا قرار دادم.

```
data['tokens_list']=data['tokens'].apply(lambda x:re.sub('[\[|\]| ]','',x).split(',') if not pd.isna(x) else np.NaN)
```

حل مسائل

مسئله اول : خطا ها رو پیدا کن

مورد اول:

در صورت مسئله گفته شده بود که همه‌ی داده ها مربوط به یک روز مشخص هستن در صورتی که بعد از بررسی که انجام دادم متوجه شدم ۱۲۳ رکورد این داده ها مربوط به روز های بعد و یا قبل هستن. توی جدول زیر دو مورد از این داده ها رو نشون دادم.

index	action	created_at	source_event_id	device_id	post_page_offset	tokens	post_index_in_post_list	post_token	tokens_list
26178	load_post_page	2021-01-01 04:06:09.213	1fc5ef66-aaad-4cf5-8e36-1e974de66fdf	lJgu0b3FQG6_grFbXenl2Q	0.0	[wXtbOKF-, wXtLeB6p, wXpbfeIJ, wXt38PoX, wXt7K...	NaN	NaN	[wXtbOKF-, wXtLeB6p, wXpbfeIJ, wXt38PoX, wXt7K...

index	action	created_at	source_event_id	device_id	post_page_offset	tokens	post_index_in_post_list	post_token	tokens_list
22088	click_post	2021-01-03 03:58:32.004	86dd16f9-8356-4a5b-8ad2-be903db7b60a	Q_Ygkm1cSDaU7_NsoUcA9w	NaN	NaN	65.0	wXvjHt4X	NaN

برای پیدا کردن این داده ها بعد از کمی جست و جو توی دیتا متوجه شدم اکثر داده ها مربوط به تاریخ ۲۰۲۱/۰۱/۰۲ هستند و بعد با کد زیر یک ماسک درست کردم که رکوردهایی که مربوط به این تاریخ نیستن رو برام مشخص کنه.

```
jan2=data['created_at'].apply(lambda x:x.day==2 and x.month==1 and x.year==2021)
```

البته توی این مسائل چون با زمان انجام اکشن توسط کاربر سر و کار نداریم این خطا مشکلی برامون به وجود نمیاره و لازم نیست از دیتا حذفش کنیم.

مورد دوم:

در ۱۴۸۷ رکورد مقدار `device_id` برابر `NaN` هست به این معنا که برای این رکورد ها device_id مشخص نیست. مشخص نبودن device_id برای یک رکورد در مسئله های پیش رو اهمیتی نداره. اگرچه اگر هم مهم بود تنها ۱.۳۳ درصد رکوردها این مشکل رو داشتند و با حذفشون مشکلی پیش نمیومد.

برای اینکه این گزارش جمع و جور و خلاصه باشه سعی میکنم فقط جدول هایی که به نظرم مهم هستن بیارن و این جدول ها به صورت کاملتر توی نوت بوکی که همراه این فایل هست، قابل مشاهدست

مورد سوم:

۶۷ رکورد تکراری در دیتا وجود داره. به این معنی که ۶۷ داده دیگر دقیقا مشابه این رکورد ها وجود داره و کاربر نمیتونه در یک زمان ثابت دو اکشن یکسان انجام بده. این داده های تکراری به صورت دو تایی هستند. یعنی از یک رکورد، یک رکورد تکراری داریم و نه بیشتر.

از اون جایی که تعداد این رکوردها خیلی ناچیز هستن ارزش نداره که بخوایم این تعداد رو از داده ها پاک کنیم. اما بهتره انجام بشه.

مورد چهارم:

با توجه به این که این داده ها مربوط به سرچ هایی هست که کاربران در اپلیکیشن انجام دادند، وقتی کاربر روی یک پست کلیک میکنه باید از قبل اون پست رو لود کرده باشه پس اگر در یک سرچ (در یک `source_event_id`) اکشن `click_post` داریم حتما باید `load_post_page` هم داشته باشیم. اما در تعداد زیادی از source_event_id ها اینطور نیست.

اما چطور این موارد پیدا شدن؟

اول داده ها رو بر اساس source_event_id یکسان گروه کردم و بعد بررسی کردم که ایا موردی هست که توی اون `click_post` باشه اما `load_post_page` نباشه. و از این موارد یک ماسک درست کردم و در اخر ۴۹۱۵ `source_event_id` پیدا شد که این مشخصات رو دارن. در تکه کد زیر نحوه استفاده از تابع `groupby` و `agg` برای انجام این کار نشان داده شده.

```
click_no_load_gp=query_grouper(data).agg({'action':lambda x:x.isin(['click_post']).any() and not x.isin(['load_post_page']).any()})
```

مورد پنجم:

در زمانی که در یک سرچ `click_post` داریم و `load_post_page` هم داریم، وقتی روی یک پست کلیک شده حتما باید قبلا در لیست پست های لود شده باشه. اما در ۴۱۳ `source_event_id` اینطور نبوده.

برای پیدا کردن این موارد بعد از گروه کردن داده هایی که در اون ها هم `click_post` بوده هم `load_post_page`، تمام توکن های کلیک شده و تمام توکن های لود شده در یک لیست قرار دادم و بعد، بررسی کردم که ایا موردی بوده که پست کلیک شده در پست های لود شده نباشه.

این مورد رو به علت اینکه کد طولانی تری داشت در این توضیحات نیاوردم اما به صورت کامل در نوت بوک هست.

- **موارد چهارم و پنجم خطاهای قابل چشم پوشی نبودند و در ادامه برای متریک هایی که لازم بود، این `source_event_id` ها از داده ها حذف شدند**
- **البته ممکنه این موارد خطا به حساب نیان و مثلا کاربر روی پستی که ذخیره کرده کلیک کرده باشه یا روی پستی که روی صفحه اول دیوار هست، اما برای تحلیل ما که روی سرچ هست، خطا هستن و باعث ایجاد خطا در نتایج و تصمیم گیری ها میشن.**

مسئله دوم: محاسبه دو متریک

dark query percent:

اینجا هدف پیدا کردن کوئری هایبه که کمتر از ۱۰ نتیجه برای اونها نشون داده شده.

برای پیدا کردن این موارد اول داده هایی که توی اون ها ستون `action` فقط شامل `load_post_page` میشد رو جدا کردم چون اینجا فقط با این اکشن کار داریم

```
load_post=data[data['action']=='load_post_page']
```

بعد اون ها رو بر اساس `source_event_id` گروه کردم و ستون tokens_list رو لیستی از توکن تمام پست های لود شده قرار دادم.

```
def tokens_lister(tokens_series):
    # Filter non-NaN series
    tokens_series_not_nan=tokens_series[~pd.isna(tokens_series)]
    tokens_list=[]
    # Add each token in tokens_series to tokens_list
    for tokens in tokens_series_not_nan:
        tokens_list+=tokens
    return tokens_list
dark_query_gp=query_grouper(load_post).agg({'tokens_list':tokens_lister})
```

و در اخر هم کوئری هایی که طول این لیستشون کمتر از ۱۰ بود رو محاسبه کردم و تعداشون رو به تعداد کل کوئری های unique تقسیم کردم.

که در اخر درصد کوئری های dark برابر **۷.۹۶** شد.

query bounce rate:

برای پیدا کردن کوئری هایی که کاربر روی هیچ کدوم از نتایج اون ها کلیک نکرده نباید کوئری هایی که در خطاهای چهارم و پنجم به دست اومدن رو در نظر بگیریم. چون همونطور که گفته شد این به نظر نمیرسه این کوئری ها مربوط به سرچ باشند چون با منطق سرچ جور در نمیان.

برای پیدا کردن این کوئری های بدون کلیک روی داده های بدون ارور، ابتدا داده ها رو بر اساس `source_event_id` گروه کردم و چک کردم که ایا توی هر گروه (کوئری)، حداقل یک `action` با نام `click_post` وجود داره یا نه و بر این اساس تعدادشون رو محاسبه کردم.

```
data_no_click_err=data[data['source_event_id'].isin(click_no_err)]

query_bounce_gp=query_grouper(data_no_click_err).agg({'action':lambda x:x.isin(['click_post']).any()})

query_bounce_len=len(query_bounce_gp[query_bounce_gp['action']==False])
```

که در آخر مشخص شد در **۳۸.۷۶** درصد کوئری های بدون خطا، کلیکی انجام نشده.

مسئله سوم: کدوم متریک مناسب تره؟

برای حل این مسئله و برای این که کارم در ادامه راحت تر بشه و درصورت نیاز بتونم به هر کدوم از این متریک ها برای هر کوئری دسترسی داشته باشم یک دیتا فریم ساختم که این ۴ متریک رو برای همه کوئری های بدون ارور نمایش بده. دلیل اینکه چرا داده های بدون ارور رو جدا کردم در بالا توضیح دادم.
کد این قسمت خیلی طولانیه و به همراه کامنت در نوت بوک موجود هست.

کدام متریک مناسب تره؟:
متریک های دوم تا چهارم همگی به این اشاره دارن که مهمه که کاربر **زودتر** روی یکی از اگهی های کوئری کلیک کنه. اما من فکر میکنم با توجه به اینکه دیوار آگهی‌ها رو به ترتیب زمان نشون میده این مهمتره که کاربر روی چند آگهی کلیک کرده و **متریک مناسب درصد آگهی های کلیک شده نسبت به آگهی های لود شده هست** . چون ممکنه کاربر حتی آگهی (های) مورد نظرش رو در لود پست های پایین تری پیدا کنه اما همون اگهی مورد نظرش باشه.
اما برای درک بهتر دو سناریو رو بررسی میکنم:

- **اگر در یک کوئری کاربر روی هیچ آگهی کلیک نکند** میتوان نتیجه گرفت این کوئری متناسب با خواسته کاربر نبوده که این موضوع در متریک های اول و دوم و سوم منعکس خواهد شد.
- **اگر هم تعداد زیادی کلیک داشته باشد** میتوان نتیجه گرفت که نتایج برای کاربر مناسب بوده و به دنبال جزئیات آگهی ها برای انتخاب دقیق تر بوده که این موضوع فقط در متریک اول منعکس شده است. و میشه نتیجه گرفت متریک مناسب متریک اول هست.

اما به نظر من متریک مناسب تر، متریکی است که میزان تعامل مخاطب با پست های کلیک شده را مشخص کند. برای مثال فرمولی طراحی کنیم که به مجموع میزان زمان صرف شده روی یک آگهی کلیک شده (به صورت جدا برای آگهی های بدون عکس و عکس دار، زیرا به طور کلی زمان صرف شده برای عکس دار ها باید بیشتر باشد) و کلیک بر روی گزینه تماس و یا چت امتیاز دهیم و این امتیاز را به تعداد پست های کلیک شده تقسیم کنیم. و این متریک را برای کوئری های بدون کلیک صفر قرار دهیم. به نظر من محاسبه میانگین این متریک برای همه کوئری ها معیار مناسب تری برای متوجه شدن میزان مناسب بودن کوئری ها برای کاربران هست. چون ممکنه یک کاربر آگهی مد نظر خودش رو در صفحه ۶ ام کوئری پیدا کنه و در طول کل کوئری هم فقط روی اون کلیک کرده باشه. این کوئری و آگهی هاش مطابق خواسته کاربر بوده اما در همه متریک های بالا امتیاز کمی کسب میکنه.

در آخر میانگین متریک اول برای همه کوئری های بدون خطا **۹.۱۲ درصد** شد. میتوان اینطور تفسیر کرد که به طور میانگین کاربران بر روی ۹ درصد پست های لود شده کلیک میکنن.
اما برای اینکه درک بهتری از این متریک پیدا کنیم جدول زیر اطلاعات کامل تری دراختیار ما قرار میده.

count	mean	std	min	25%	50%	75%	max
8968.000000	9.127546	19.592342	0.000000	0.000000	2.777778	8.898810	300.000000

از این جدول میتونیم متوجه بشیم که ۷۵ درصد کوئری ها rate کمتر از ۸.۸۹ درصد دارن.
اما درصد کمی از کوئری ها وجود دارن که کاربر حتی بیشتر از تعداد آگهی لود شده کلیک کرده و این بسیار عالیه. اما احتمالا خیلی کمتر از ۲۵ درصد کوئری ها این ویژگی رو دارن.

مسئله چهارم: پرتاب سکه برای کلیک روی آگهی!

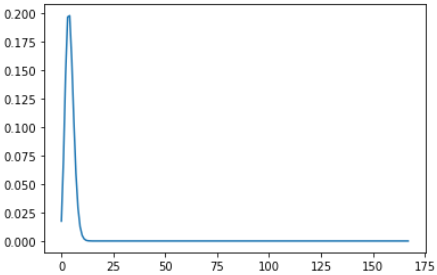
توزیع برنولی که یک توزیع گسسته است که فقط دو مقدار برای متغیر تصادفی X میپذیرد (۰ و ۱ یا پیروزی یا شکست). در این جا میتوانیم در هر کوئری، هر آگهی را یک آزمایش برنولی در نظر بگیریم که اگر کاربر روی آن کلیک کند موفق است و اگر کلیک نکند نا موفق است. برای اینکه شهود بهتری داشته باشیم، میتوان گفت کاربر هر بار با دیدن آگهی یک تاس می اندازد که اگر شیر بیاید کلیک میکند و اگر نه کلیک نمیکند. این آگهی ها از هم مستقل اند و میتوان مقدار احتمال موفقیت (p) رو نسبت کلیک ها در یک کوئری به کل آگهی ها در نظر گرفت.

از اون جا که در هر کوئری تعداد مشخصی (n) پست داریم و در واقع به همان تعداد آزمایش برنولی داریم. پس میتوان کل کلیک های یک کوئری را با توزیع دو جمله ای مدل کرد که p را هم همان نسبت کلیک ها در نظر میگیریم.
باید بگم این مقداری که برای p در نظر گرفته شده قطعا بهترین مقدار و مقدار دقیقی نیست اما با توجه به داده هایی که در دست داریم بهترین انتخابه.
با داشتن n و p مدل دو جمله ای ما اماده است. برای اینکه دسترسی بهتری به این پارامتر ها در هر کوئری داشته باشیم یک دیتا فریم درست کردم که این مقادیر رو برای هر کوئری در اختیارمون بذاره. که قسمتی از این دیتافریم رو در زیر آوردم.

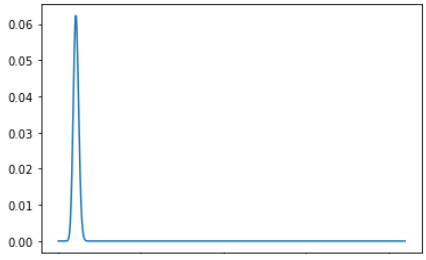
source_event_id	Probability of succsess (p)	Number of trials (n)
ffda8068-9c22-46f0-971e-6b2b19c764ea	0.009259	216
ffeaa17f-ee79-4f42-be58-afd60a5aef37	0.000000	48
ffec4e11-c0a1-4fa4-8613-0979d6f46918	0.009470	528

به کمک توزیع دو جمله ای میتونیم بررسی کنیم که احتمال کلیک های بیشتر و یا کمتر در این کوئری چه مقدار است و برای هر مورد دلخواه نمودار توزیع احتمال رو برای میزان موفقیت های مختلف رسم کنیم.

برای مثال در توزیع زیر که مربوط به سومین داده جدول است رو میشه مشاهده کرد که احتمال داره که کاربر در این کوئری هیچ کلیکی انجام نده. چون احتمال آن کمی کمتر از ۲ درصد است.



اما در توزیع زیر این احتمال تقریبا صفر است و میتوان گفت کاربر در همچین سرچی به احتمال زیاد کلیک انجام میدهد و این تحلیل در تحلیل کوئری ها به ما کمک خواهد کرد.



آیا با استفاده از این توزیع میتوان با در دست داشتن یک متریک تخمینی از متریک های دیگر محاسبه کرد؟

توزیع دوجمله‌ای (و همینطور حالت خاص آن، توزیع برنولی) فقط به شکست یا پیروزی و تعداد آن در طول آزمایش ها اهمیت میدهند. و برای آن ها مهم نیست این پیروزی یا شکست در چه زمانی و با چه ترتیبی اتفاق می‌افتد. **بنابراین در هیچ حالتی نمیتوان تخمینی از متریک های دوم و سوم (رتبه اولین کلیک و میانگین رتبه کلیک ها) به دست آورد.**

اما مقداری که برای p در نظر گرفتیم همون متریک شماره یک هست. و با در دست داشتن اون میتونیم تخمینی از متریک شماره چهار (ایا روی سه پست اول کلیک شده یا نه) داشته باشیم. اما چطور؟

هر کلیک روی آگهی یک آزمایش برنولی با p=metric$_{۱/۱۰۰}$ هست. و برای اینکه احتمال کلیک شدن روی یکی از سه پست اول هر کوئری رو بررسی کنیم باید عبارت زیر رو محاسبه کنیم.

`p=probability of sucess`

`q=probability of failure=1-p`

`p1=probability of one sucess in first three trial`

$$p1 = (p*p*p + p*p*q + p*q*q) * \sum_{i=0}^{n-3} \left(p^i * q^{n-3-i}\right)$$

پیاده سازی این عبارت در تابع `any_of_first_3` انجام دادم که با دریافت p و n، اگر احتمال p۱ بیشتر از ۵۰ درصد باشد true و در غیر این صورت false بر میگرداند.

```
def any_of_first_3(n,p):
   n=int(n)
   m=0
   q=1-p
   for i in range(n-3+1):
        m+=p**i+q**(n-3-i)
   return p*q*q*m+p*p*q*m+p*p*p*m > 0.5
```

بعد از این و با محاسبه این مقدار برای تمام کوئری ها مقدار تخمین زده شده متریک سه رو در یکی از ستون های دیتا فریم `metrics` گذاشتم و بعد از مقایسه متوجه شدم این تخمین دقت **۶۶.۱۹** درصدی داره که برای یک مدل آماری قابل قبوله.

اما حالت عکس این تخمین امکان پذیر نیست چون در حالت عکس اگر مقدار true رو برابر ۱ و مقدار false رو برابر ۰ در نظر بگیریم برای رسیدن به مقدار p باید یک معادله درجه ۲ رو حل کنیم که در صورتی که به دو ریشه مثبت حقیقی برسیم هیچ معیاری برای تشخیص مقدار درست نداریم و بنابراین این تخمین امکان پذیر نیست.

پس مشخص شد فقط با داشتن متریک شماره ۱ میتوان تخمینی از متریک شماره ۴ ارائه داد.