به نام خدا

درس: مقدمه‌ای بر یادگیری ماشین

استاد: دکتر صالح کلیبر

گزارش تمرین شماره ۲

سیّدمحمّدامین منصوری طهرانی

۹۴۱۰۵۱۷۴

توجه ۱: در کد پیوست شده، تعدادی تابع برای اجرای خواسته‌های مختلف سوالات نوشته شده و قسمت اصلی کد به صورت کامل کامنت شده است. برای مشاهده خروجی هر بخش لطفاً همان‌طور که برای هر بخش اشاره شده، یک یا چند خط مربوط به آن را از حالت کامنت خارج کنید و کد را اجرا کنید. این کار برای جلوگیری از اجرا شدن همه بخش‌ها با هم و احیاناً crash کردن برنامه انجام شده است.

توجه ۲: لطفاً در طرح تمرین‌ها مقداری نرم‌تر عمل کنید! حجم کاری این تمرین واقعاً زیاد بود!

توجه ۳: اجرای روش برچسب زنی ۷ ساعت طول می‌کشد و برای همین این قسمت هم به صورت خودکار انجام نمی‌شود. باید کد این بخش را uncomment نمایید تا روند را مشاهده کنید.

توجه ۴: لطفاً فایل‌های داده را برای اجرای صحیح کد در مسیری که فایل را قرار می‌دهید بگذارید.

فایل متنی search-result.txt:

دیتافریم train به صورت متغیر global تعریف شده و این اتفاق مکرراً اتفاق افتاده تا فرآیند load کردن سنگین آن فقط یک بار باشد و در سایر مکان‌ها استفاده شود. سپس برای جمع‌آوری سطرهای مربوط به یک کاربر در تمامی فایل‌ها، فایل users.csv نیز خوانده می‌شود. در ورودی این تابع شماره یک کاربر نیز گرفته می‌شود و این شماره طبیعتاً دلخواه است و برای سادگی اولین کاربر را انتخاب کردم. سپس در هر دو فایل در ستون user_id جستجو کرده و اگر شماره آن با کاربر انتخابی ما یکسان بود همه داده‌های آن را به متغیر جدیدمان اضافه می‌کنیم و در نهایت این اطلاعات را در یک فایل txt ذخیره کردم. برای هر کاربر تعداد نوتیفیکیشن‌ها می‌تواند بیشتر از یکی باشد و سطرهای فایل پیوست شده همین را نشان می‌دهد. بنابراین ابتدای فایل مربوط به مشخصات واکنش کاربر نسبت به نوتیفیکیشن در فایل train.csv و پس از آن ویژگی‌های این کاربر در فایل users.csv است.

---

۳ . ۴ کار با داده‌های گم‌شده:

با توجه به قرار گرفتن این داده‌ها به طرز بسیار نامرتب، با برداشتن کامنت این قسمت، تابع آن به شرح زیر عمل می‌کند. ابتدا درصد داده‌های گم‌شده را در هر ستون داده users پیدا می‌کند. اگر ستونی بیشتر از ۱۵ درصد داده گم‌شده داشت، آن را حذف می‌کند. سپس هر سطری که در آن یک داده گم‌شده یافت شود، آن سطر را به طور کلی حذف می‌کند. دقت کنید اگر ابتدا سطرهای حاوی داده گم‌شده را حذف می‌کردیم، تعداد زیادی داده حذف می‌شد(به خاطر ستون‌های حاوی داده گم‌شده زیاد). امّا با حذف آن ستون‌ها، داده‌های مفید بیشتری نگه داشته می‌شوند. خروجی این تابع، داده users ای است که داده‌های گم‌شده آن حذف شده‌اند. یک خروجی دیگر آن هم درصد داده‌های گم شده هر ستون است که می‌توانید آن‌را ملاحظه کنید. توجیه کاری که در این قسمت کردیم این است که برای جای‌گذاری داده‌های گم‌شده با توجه به این‌که از ماهیت ستون‌ها اطلاعی نداریم، تصمیم درست و متقنی برای محاسبه آن‌ها بر اساس داده‌های همسایه نخواهیم داشت، از طرفی حجم داده بسیار بزرگ است و ویژگی‌های متنوعی برای کار کردن وجود دارد و از آن‌جا که سادگی پارامتری مهم می‌باشد و نسبت داده‌های حذف شده به کل داده‌ها نسبتاً ناچیز است، به نظر معقول و منطقی می‌نماید که داده‌های گم‌شده و سطرهای حاوی آن‌ها به طور کلی حذف شوند و به نظر نمی‌رسد لطمه‌ای به مسأله بخورد زیرا همان‌طور که تأکید شد، از ماهیت داده‌ها اطلاعی در دست نیست و نمی‌توان برای درون‌یابی آن‌ها از روش مطمئنی استفاده کرد.

---

۳ . ۵ نشتی اطلاعات

برای جلوگیری از نشتی اطلاعات باید به تعریف آن دقت کرد و مدل را براساس اطلاعاتی که بعد از ثبت نتیجه بدست آمده‌اند نسازیم؛ در این مسأله چند زمان وجود دارد، یکی زمان ارسال است که به اپلیکیشن مربوط است و ما آن‌را می‌دانیم(زمانی است که خودمان عملیات ارسال را انجام داده‌ایم.). یک زمان هم زمان رسیدن یا delivery پیام به کاربر است که با روشن شدن Wi-Fi کاربر یا mobile data او، اپلیکیشن می‌تواند پیامی که مخابره شده بود را به کاربر نشان دهد و زمان این نشان دادن را نیز برای ما بفرستد پس این زمان هم نشتی اطلاعات ندارد و می‌توان از آن برای ساخت مدل استفاده کرد. اما زمان تعامل کاربر یا زمان interaction او نشتی اطلاعات دارد زیرا پس از عمل کاربر مشخص می‌شود و در واقع طبق تعریف، اپلیکیشن پس از ثبت نتیجه (notification accept or reject) این اطلاعات را در اختیار ما می‌گذارد و ما مجاز به استفاده از این زمان در ساخت مدل پیش‌بینی خود نیستیم. پس برای اطمینان از صحت مدل و محاسبات آینده، این ۳ ستون را با تابعی که پس از uncomment کردن آن و اجرای کد عمل می‌کند، از فایل train حذف می‌کنیم.

---

۴ تحلیل اکتشافی داده

۱.۴ آماره‌های مهم

برای چاپ شدن آماره‌های روز و دقیقه و ساعت نوتیفیکیشن‌ها برای همه کاربرها، لطفاً خط مربوط را از حالت comment در بیاورید و کد را اجرا کنید. در تابع print_statistics ابتدا از متغیر global ای که قبلاً هم اشاره کردیم استفاده می‌کنیم و ستون‌های مختلف داده train، یعنی روز و ساعت و دقیقه را جدا می‌کنیم و برای هر ستون، کمیت‌های میانگین و مد و واریانس و میانه را چاپ می‌کنیم. نتیجه این تابع در زیر آورده شده است.

```
Python Console
>>> runfile('/home/amin/PycharmProjects/ML_HW02/Main.py', wdir='/home/amin/PycharmProjects/ML_HW02')
The average hour a notification arrives is: 13.23668735
The mode of hour a notification arrives is: 15
The median of hour a notification arrives is: 14.0
The variance of hour a notification arrives is: 21.11792110424595
The average minute of minute a notification arrives is: 29.51711695
The mode of minute a notification arrives is: 30
The median of minute a notification arrives is: 31.0
The variance of minute a notification arrives is: 286.71973934600896
The average day a notification has arrived is: 3.7301379
The mode of day a notification arrives is: 1
The median of day a notification arrives is: 3.0
The variance of day a notification arrives is: 4.194197956693494
```

تحلیل خروجی فوق بدون در نظر داشتن زمان تعامل‌ها و رد یا قبول شدن آن، تحلیل چندان مفیدی نخواهد بود. میانگین روزی که پیام‌ها رسیده‌اند روز اول است و بیشترین پیام‌ها در روز اول رسیده‌اند و این می‌تواند ناشی از تقارن این روز با مناسبتی بوده باشد و در این روز تعداد پیام‌های اپلیکیشن‌ها به مناسبت این روز(تخفیف و ...) زیاد شده باشد. زیاد شدن پیام‌های مناسبتی نیز می‌تواند احتمالاً جواب مثبت گرفتن آن‌ها از کاربر را کاهش دهد. روز میانه هم به روز میانگین نزدیک است که نشان می‌دهد توزیع رسیدن پیام‌ها در اطراف میانگین نسبتاً یکسان است و دارای اریب نمی‌باشد.(Skewness) انحراف معیار دو روز نیز نشان می‌دهد اکثر پیام‌ها در فاصله ۲ روز از میانگین ارسال شده‌اند. در مورد دقیقه وضعیت تحلیل بدتر است زیرا دلیلی برای ارسال پیام در دقیقه‌های مختلف ساعت وجود ندارد. به عبارت دیگر بعید است دقایق یک ساعت ارجحیت خاصی نسبت به هم داشته باشند که اپلیکیشن بخواهد در دقیقه‌ای پیام بیشتری ارسال کند! تحلیل فوق با نتیجه که میانگین نیز نزدیک وسط بازه (دقیقه ۳۰) است هم‌خوانی دارد. میانه و میانگین بازهم نزدیک هستند که نشان از اریب نداشتن توزیع رسیدن پیام‌ها حول میانگین دارد. انحراف معیار ۱۷ دقیقه هم توزیعی شبیه نرمال حول میانگین را به ذهن می‌آورد. اما بیشترین پیام‌ها در دقیقه ۳۰ رسیده‌اند که این می‌تواند ناشی از فعال شدن تایمر آن‌ها در هر ۳۰ دقیقه برای ارسال پیام باشد و به این ترتیب در وسط بازه هر ساعت اپلیکیشن‌های زیادی پیام بفرستند و این مد را ایجاد کرده باشند.

در مورد ساعت به نظر می‌رسد میانگین زمان رسیدن در ظهر باشد که دور از انتظار هم نیست. البته انتظار عصر هم می‌رفت. بیشترین زمان رسیدن در ساعت ۳ بعد از ظهر بوده است و شاید علت این تعداد بالای صفر همین

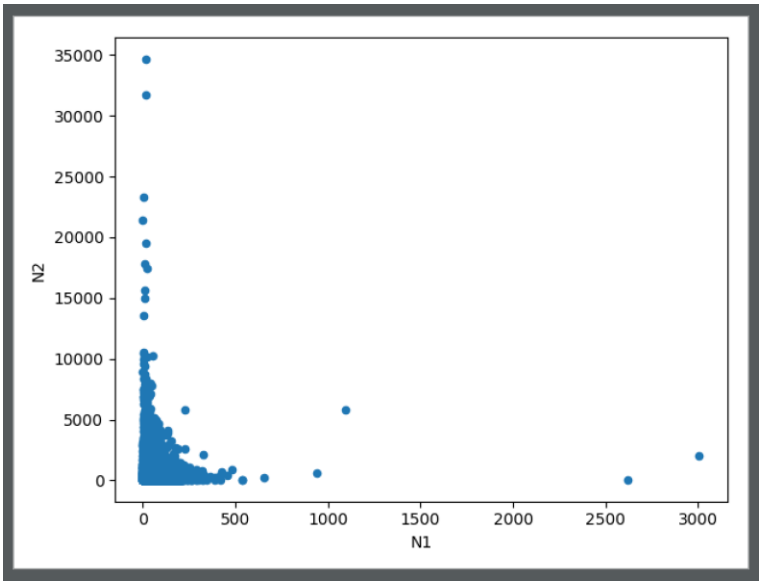
زمان بد فرستادن اعلان‌ها باشد! (مقارن با زمان خواب و استراحت!) میانگین و میانه نیز دوباره نزدیک هم هستند که نتیجه مشابه حالت‌های قبل را بدست می‌دهند. انحراف از معیار هم نسبتاً زیاد است و به نظر می‌رسد اکثریت پیام‌ها از بازه صبح تا عصر ارسال شده‌اند.

در ادامه کد با uncomment کردن کد بعدی در تابع statistics_calculator، میانگین و میانه و واریانس به ازای هر کاربر (با دستور groupby) محاسبه شده است و در خروجی بازگردانده شده‌است. از آن‌جایی که تعداد کاربران بسیار زیاد است و سوال هم چیزی در مورد تحلیل این داده‌ها نگفته، به نظر نمی‌رسد انتظاری برای تحلیل این حجم از داده وجود داشته باشد!

برای محاسبه مد، در هر ستون از داده‌های مورد نظر که با دستور groupby مرتب شده‌اند، تعداد اعداد مختلف یکتا شمرده شده و ماکزیمم آن‌ها به عنوان مد برگردانده می‌شود.

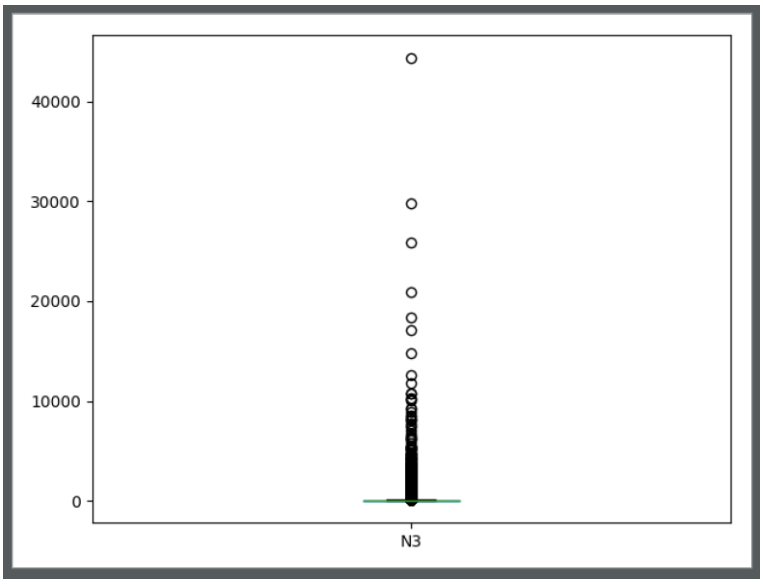
۴ . ۲ نمودارهای مهم (لطفاً کد مربوط به هر قسمت را uncomment کنید و سپس اجرا نمایید.)

نمودار پراکندگی:



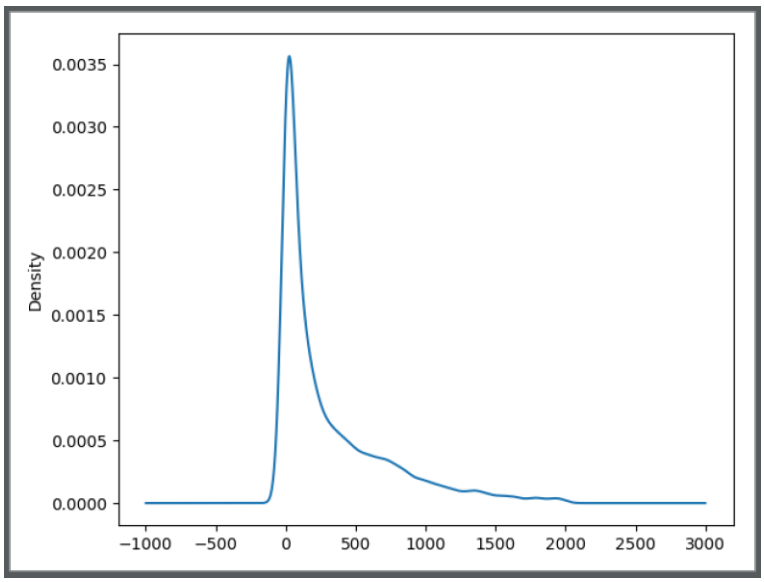
این‌ها ویژگی‌های عددی کاربران هستند و مشاهده می‌شود مقادیر نوعی یکی ۱۰ برابر دیگری بوده و محدوده‌ای که مقادیر به صورت معنادار وجود دارند (داده پرت نیستند) برای ویژگی N2 تا ۲۰ برابر محدوده معنی دار ویژگی N1 نیز می‌رود. هم‌چنین مشخص می‌شود که این دو ویژگی کاربران را از هم جدا نکرده و احتمالاً اطلاعات چندانی در اختیار ما نخواهد گذاشت و ضرر زیادی در طی استفاده نکردن از آن‌ها برای ساخت مدل متحمل نخواهیم شد. چون همه کاربران در محدوده تفکیک ناپذیری در نزدیکی مبدأ در فضای این دو متغیر قرار دارند.

نمودار جعبه‌ای:



در نمودار روبرو می‌بینیم که قسمت چارک اول و سوم ناحیه بسیار کوچکی را به خود اختصاص داده است (مستطیل بسیار کوچک سبزرنگ) و دایره‌های توخالی داده‌های پرت یا outlier ها را نشان می‌دهند. خط سیاه وسط نیز چارک دوم یا میانه است. چارک اول جایی است که یک چهارم داده‌هایی که کمتر از میانگین هستند در آن قرار دارند و چارک بالا نیز به همین ترتیب تعریف می‌شود. بنابراین برای کار با این ویژگی باید در نظر داشت که اکثر اطلاعات در ناحیه نزدیک به صفر قرار گرفته و باید داده‌های پرت را حذف نمود و سپس از این ویژگی استفاده کرد. البته با توجه به ناحیه کمی که به اکثر داده‌ها مربوط است (ناحیه سبزرنگ)، به نظر می‌رسد واریانس آن‌ها کم بوده و ممکن است این ستون هم برای استفاده در ساخت مدل کمک چندانی نکند.

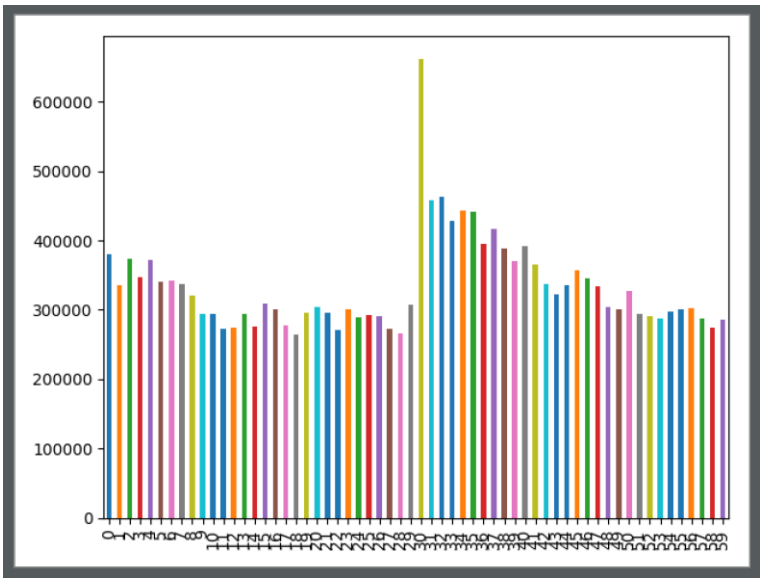
نمودار توزیع:



در تابع مربوط به این قسمت (distribution_plotter) ابتدا فایل اعلان‌ها لود شده و با دستورهایی که مشاهده می‌شود به دیتافریم بزرگی که تعداد ستون‌های آن به گونه‌ای تنظیم شده که هر کلمه در یک ستون قرار بگیرد تبدیل می‌شود. اگر سطری کم‌تر از این تعداد کلمه داشته باشد خانه آن None خواهد شد. سپس در تمام خانه‌های این دیتافریم بزرگ (از جایی به بعد که به کلمات اعلان‌ها مربوط است) جستجو می‌کنیم و اگر None بود، به سراغ کلمه بعدی می‌رویم. در غیر این‌صورت این عدد را به لیستی اضافه می‌کنیم. این لیست به سری تبدیل شده و برای اعداد داخل آن با دستوری که دیده می‌شود توزیع رسم می‌شود. نتیجه تصویر بالا خواهد بود. در تصویر مشاهده می‌شود که کلماتی که در دیکشنری مقادیر کمی دارند (تا ۲۰۰) مکرراً استفاده می‌شوند و احتمالاً کلمات پر تکراری چون

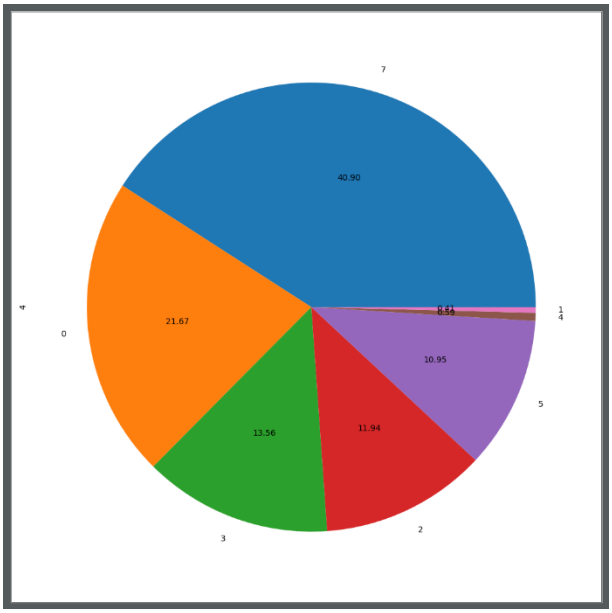
{سلام} و یا کلمات مشابه باشند که در تصمیم تأثیری ندارند. کلمات با اندیس‌های متوسط تعداد تکرار کمتری دارند و احتمالاً این کلمات تعیین کننده نوع اعلان هستند و براساس این‌ها می‌توان مدل را بهبود بخشید.

نمودار میله‌ای:



در تابع این قسمت ستون دقیقه برداشته شده و مقادیر یکتای آن شمرده می‌شوند و با دستور مناسب نمودار میله‌ای آن رسم می‌شود. مشابه تحلیل‌هایی که در قسمت‌های قبل داشتیم، بیشترین دقیقه رسیدن اعلان یا همان مد دقیقه ۳۰ است که احتمالاً به زمان‌بندی اعلان‌ها رأس ساعت‌های خاصی برمی‌گردد ولی توزیع رسیدن اعلان‌ها به جز در مرکز، کم و بیش یکنواخت به نظر می‌رسد و این دور از انتظار هم نیست زیرا بین دقیقه‌های مختلف ساعت برای اپلیکشینی ترجیحی برای ارسال اعلان وجود ندارد و از نمودار میله‌ای فوق نیز می‌توان نتیجه قبلی را مبنی بر مفید نبودن دقیقه ارسال یا رسیدن اعلان در یک ساعت به کاربر برای ساخت مدل پیش‌بینی تأیید کرد. در واقع منطقی نیست در این متغیر اطلاعات معناداری نهفته باشد.

نمودار دایره‌ای:



با اجرای تابع این قسمت نیز فایل نوتیفیکیشن‌ها لود شده و خوانده می‌شود و به ۶ ستون تقسیم می‌شود که چهارمین آن‌ها همان ویژگی دسته‌ای است و ما آن‌را استخراج می‌کنیم. سپس آن‌را از رشته متنی به عدد تبدیل می‌کنیم و مقادیر یکتا را می‌شماریم و با دستور مربوط نمودار خواسته شده را رسم می‌کنیم. شکل روبرو حاصل می‌شود. مشاهده می‌شود که درصد زیادی از پیام‌ها از نوع ۷ هستند. به طور کلی مشاهده می‌شود ویژگی پیام‌ها به خوبی از هم جدا شده و به تعداد دسته‌های معقولی تقسیم شده و می‌توان از آن‌ها برای دسته‌بندی و ساخت مدل استفاده کرد. احتمالاً هر دسته مفهوم یا موضوعی خاص را اعلام می‌کند. امّا مشکل اساسی این است که در داده‌های تست، notif_id ها جدید هستند! بنابراین هیچ اطلاعی از ویژگی‌های آن‌ها نداریم و

اگر مدلی بر اساس این ویژگی‌ها بسازیم، برای تست دست‌آویزی نخواهیم داشت و با شکست مواجه خواهیم شد. بنابراین علی‌رغم این‌که این ویژگی مؤثر و مفید به نظر می‌رسد، با توجه به نداشتن ویژگی اعلان‌های جدید بلا استفاده خواهد ماند و ما از آن در ساخت مدل استفاده نمی‌کنیم. امّا اگر در مسأله‌ای دیگر ویژگی‌های همه اعلان‌ها شناخته شده بود، این روش قابل اتکا بود و حتماً از آن استفاده می‌کردیم.

---

۵ مدل‌های رگرسیونی

برای پیش‌بینی داده تست از دو روش استفاده می‌کنیم:

- روش اول: در این روش از رگرسیون استفاده نشده است.

در این روش که ساده‌انگارانه به نظر می‌رسد ولی جواب نسبتاً خوبی (البته نه با معیار خیلی بهتر بودن از حالت‌های بدیهی) می‌دهد، به این‌گونه عمل می‌شود که سابقه کاربران train بررسی شده و اگر به طور میانگین یک کاربر بیش از ۱۰ درصد اوقات کلیک کرده باشد، در داده تست برای او برچسب ۱ می‌زنیم (کلیک می‌کند) و در غیر این‌صورت برچسب او را صفر می‌زنیم. اگر هم در داده train این کاربر یافت نشود، برچسب آن‌را صفر می‌زنیم. (این انتخاب به این خاطر است که تعداد صفرها به طور فاحشی از تعداد یک‌ها بیشتر است و در این روش ساده‌انگارانه که اطلاع دیگری در دست نیست، محتمل‌تر است که این کاربران اعلان را رد کنند و برچسب آن‌ها صفر باشد.) درصد گفته شده نیز از میانگین interaction های هر کاربر بدست می‌آید. پیش‌بینی به این صورت خواهد بود که روی کاربران داده test حلقه می‌زنیم (و روند اجرای این حلقه‌ها را با کتاب‌خانه tqdm دنبال می‌کنیم) و مقدار پیش‌بینی شده برای هر کاربر را مطابق آن‌چه در بالا گفته شد قرار می‌دهیم. در انتها این لیست بزرگ را به سری تبدیل می‌کنیم و ذخیره می‌کنیم. (با توجه به زمان اجرای بسیار زیاد آن که حدود ۷ ساعت طول کشید این داده‌ها لازم است ذخیره شوند.) سپس لود می‌شود و ستون مربوط به مقادیر پیش‌بینی شده آن برداشته شده و با سیاست گفته شده به یکی از اعداد صفر یا یک نگاشته می‌شود و نهایتاً این مقادیر float به مقادیر int تبدیل می‌شوند و نهایتاً به دیتافریم داده تست اضافه می‌شوند و در فایل مورد نظر نوشته می‌شوند. این همان فایل پیوست شده به نام output.csv است. این فایل به کوئرا تحویل داده شد و خروجی آن به شرح زیر است(که مشخص می‌کند این کار قبل از ددلاین انجام شده ولی متأسفانه گزارش بعد از ددلاین آپلود شده است!):

```
Test finished successfully

    test 1
    0.9499195704568288 out of 1
    Output Report
        True Positive: 0, True Negative: 1038150, False Positive: 54732, False
Negative: 0     Accuracy: 0.9499195704568288     Precision: 0.0   Error in
calculating recall. -- Invalid numbers.
Report time: 2018-11-23 17:11:53.508728
```

```
output.zip | submission id: 1526358
```

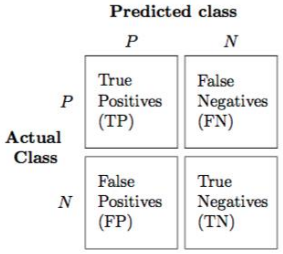
```
    Test finished successfully

    test 1
    0.9499195704568288 out of 1
    Output Report
    True Positive: 0, True Negative: 1038150, False Positive: 54732, False Negative: 0  Accuracy: 0.9499195704568288    Pre
Report time: 2018-11-23 17:11:53.508728
```

- روش دوم: در این روش با استفاده از یک ویژگی رگرسیون می‌زنیم و خروجی‌های خواسته شده را بدست می‌آوریم. این روش یک‌بار انجام شد ولی نتایج خوبی نگرفت و لذا توضیح بیشتری برای آن وجود ندارد. روش اول نتایج قابل قبول‌تری بدست داد.

۲ . ۵ معیارهای اندازه‌گیری دقت الگوریتم یادگیری

با توجه به نتایج قسمت قبل، برای روش اول ماتریس درهم‌ریختگی را بدست می‌آوریم:



$Confusion\ Matrix$:

0	0
54732	1038150

$$Accuracy = \%94.9919$$

$$Precision = \%0\ (!)$$

$$Recall = \%0\ (!)$$

$$Specificity = \frac{1038150}{1038150 + 54732} = \%94.9919$$

$$F1\ Score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = 0$$

ROC ۳.۵

از آن‌جایی که در روش اول چنین چیزی وجود ندارد، برای این بخش نیز متناظراً پاسخی وجود نخواهد داشت.

---

۶ قسمت امتیازی

در این حالت یک برچسب جدید داریم. در واقع باید مدل ما باید هر داده تست را به یکی از ۳ کلاس گفته شده بنگارد. این مسأله قابل تبدیل به طبقه‌بندی باینری است و روش‌های مختلفی برای آن وجود دارد.

- یک در مقابل همه

برای هر کلاس یک طبقه‌بند آموزش دهیم که داده‌های مربوط به کلاس خودش را ۱ و سایر داده‌ها را صفر دسته‌بندی کند و این دسته‌بندی دارای یک درجه اطمینان باشد. به این صورت هر داده جدیدی که می‌آید توسط هر ۳ دسته‌بند برچسپ زده می‌شود و یک درجه اطمینان خواهد داشت (confidence score). برچسب این داده را بیشترین درجه اطمینان تعیین می‌کند.

- یک در مقابل یک

در این حالت به ازای k دسته، از هر جفت دسته یک نمونه برداریم و یک طبقه‌بند باینری آموزش دهیم که این دو دسته را تمیز دهد. بنابراین به تعداد انتخاب ۲ از k طبقه‌بند باینری داریم. (در مسأله ما ۳ تا) به هنگام پیش‌بینی همه این طبقه‌بندهای جفت جفت بر داده جدید اعمال می‌شوند و دسته‌ای که بیشترین تعداد برچسب یک را از این طبقه‌بندهای باینری از آن خود کند، برچسب داده جدید خواهد بود.

روش‌های دیگر مانند SVM نیز می‌توانند در طبقه‌بندی چند کلاسی ما را یاری دهند. امّا روش‌های ساده‌تر بر مبنای روش‌های باینری همان‌هایی بود که در بالا به آن‌ها اشاره شد.