مباحث ویژه در سیستم‌های دیجیتال

نیم‌سال اول ۱۴۰۰-۱۴۰۱

مدرس: دکتر ایمان غلام‌پور

دانشکده مهندسی برق

مستند پروژه

نام و نام‌خانوادگی: امین کشیری

شماره دانشجویی: ۹۷۱۰۱۰۲۶

۱ مقدمه

در این پروژه، با استفاده از ایده‌ها و روش‌های مختلفی که در طول ترم آموختیم، اطلاعاتی را از داده‌ها بیرون کشیدم. بخش‌های مختلف این پروژه را در فایل‌های jupyter جداگانه قرار داده‌ام، و توضیحات هر بخش را نیز جداگانه در این مستند نوشته‌ام.

۱.۱ توضیحات کلی

۱. در ابتدای تمامی کدها تنظیمات اولیه اسپارک را انجام دادم، و سپس فایل csv داده شده را load کردم.

۲. در بعضی از بخش‌ها، روز ۸م را از داده‌ها حذف کردم. دلیل این کار این بود از تمامی روزها به اندازه‌ی متناسب با هم داده داشته باشیم. در غیر این صورت تعداد داده‌ها از روز سه شنبه دو برابر باقی روزها می‌شد. کارهای دیگری نیز می‌توانست انجام بگیرد. مثلا می‌شود داده‌های روز سه شنبه را میانگین بگیریم (یعنی در تمام قسمت‌هایی که تعداد متغییری را شمرده‌ایم، برای روز سه شنبه این تعداد را تقسیم بر ۲ کنیم). در بعضی از قسمت‌ها اما این زیادتر بودن داده‌های روز سه شنبه مشکلی ایجاد نمی‌کرد. اما دقت کنید که در بعضی از قسمت‌های دیگر می‌تواند تحلیل ما را دچار انحراف کند (مثلا ممکن است به اشتباه نتیجه بگیریم که روز سه شنبه روز پر تردد تری است، یا دوربین‌هایی که در روز سه شنبه دیده می‌شوند را به اشتباه مهم تر در نظر بگیریم).

۳. توضیحات کد و روند اجرا را در فایل‌های jupyter نوشته‌ام. سعی کرده‌ام که توضیحات منطق پشت کدها را در این مستند بنویسم (و نه در خود کدها). بنابرین توضیحات تکنیکال خود کد در اینجا کمتر نوشته شده است.

۴. در مورد بعضی از ایده‌های این پروژه با خانوم فاطمه توحیدیان با شماره دانشجویی ۹۷۱۰۰۳۵۴ بحث و گفت‌وگو کردیم،

۲ General

در اولین بخش از پروژه، سعی کردم اطلاعات اولیه‌ای را از داده بیرون بکشم، تا در بخش‌های بعدی بهتر بتوانم تحلیل کنم. کارهایی که در این بخش انجام شده است:

۱. تعداد کل داده‌ها، کل ماشین‌ها و کل دوربین‌ها را به دست آوردم. این مقادیر به ترتیب برابر با ۳۴۹۸۹۱۶۰ و ۵۴۸۷۶۴۵ و ۱۰۳۵ بودند.

۲. پر تکرارترین ماشین‌ها، و پر تکرارترین ترکیب ماشین و دوربین را به دست آوردم. در این مرحله متوجه شدم که یکی از ماشین‌ها داده‌ی پرت است (میزان ثبت شدن آن بیش از ۱ بار در هر ثانیه است). به همین دلیل در بخش‌های بعدی این داده را حذف کردم. دقت کنید که حتی اگر این داده‌پرت نباشد، و مثلا ثبت شدن‌های متوالی یک ماشین پارک شده باشد، در قسمت‌های بعدی تحلیل ما را با مشکل مواجه می‌کند. پس از نظر مفهومی حذف آن واجب به نظر می‌رسید.

۳. خلاصه‌ای از data را به دست آوردم و نمایش دادم، مانند میانگین، کمینه، بیشینه، چارک اول و سوم و ...

۴. تنها ردیف‌هایی که یک تخلف بودند را نگه داشتم، و پس از حذف داده‌های پرت، متخلف‌ترین ماشین‌ها را نمایش دادم. بیشترین تخلف مربوط به ماشین ۷۳۱۳۸۲۹۵ بود با ۷۳۰ مورد تخلف.

۵. نرخ خطای دوربین‌ها را به کمک دو ستون ORIGINE_CAR_KEY و FINAL_CAR_KEY به دست آوردم (اختلاف این دو نشان دهنده‌ی اشتباه است) و سپس به کمک آن دوربین‌ها را بر اساس دقت مرتب کردم و دقت بادقت‌ترین و کم‌دقت ترین آن‌ها را به دست آوردم.

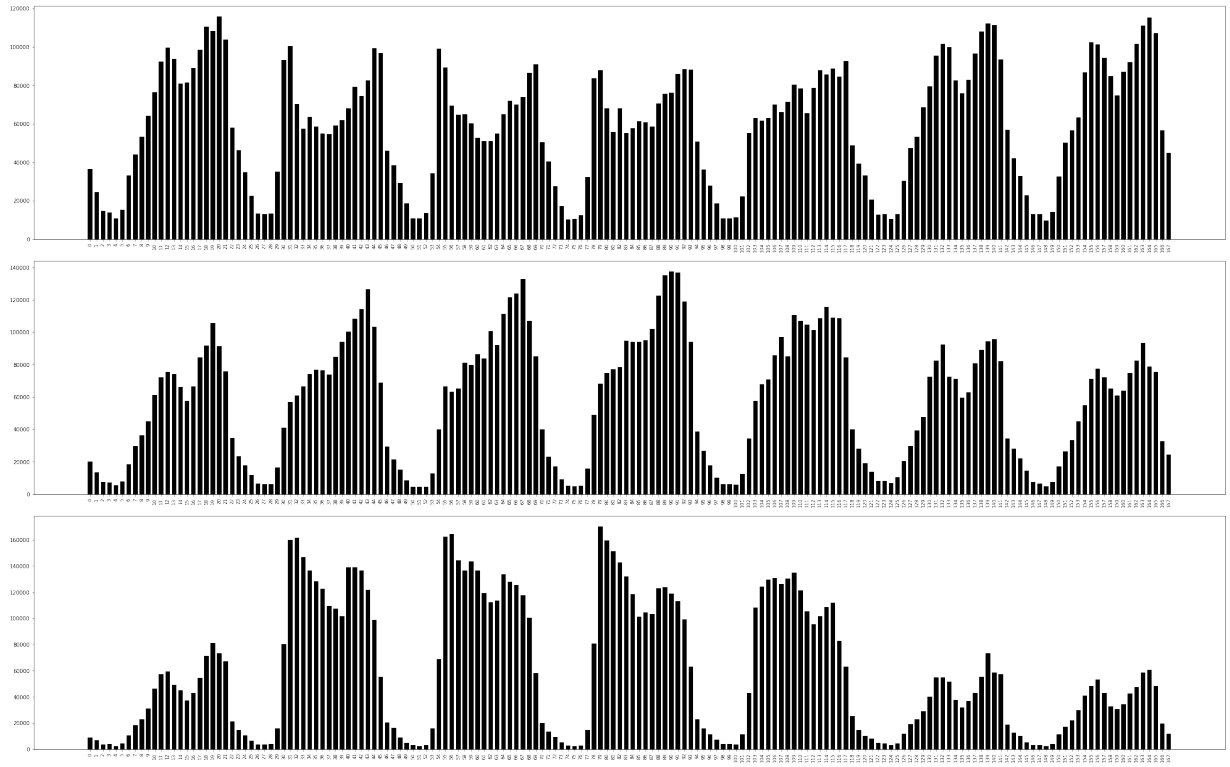
۳ Clustering

۱.۳ توضیحات کلی

با استفاده از خوشه سازی، سعی کردم دوربین‌ها را به دسته‌های مختلفی تقسیم کنم، و برای هر دسته مفهومی بیابم. نماینده هر دوربین در این روش، یک بردار با اندازه‌ی 7×24 است، که در هر خانه‌ی آن تعداد تردد در آن ساعت از روزهفته قرار گرفته‌است. ۲۴ ساعت اول برابر با یک‌شنبه است، ۲۴ ساعت بعدی برای دو شنبه و الی آخر. پس بردار متناظر هر دوربین، تعداد ترددها در هر ساعت از یک هفته را برای آن دوربین مشخص می‌کند.

برای خوشه‌سازی از الگوریتم LDA یا Latent Dirichlet Allocation استفاده شده است، که همانطوری که در درس دیدیم استفاده اولیه آن پیدا کردن توزیع topic های مختلف و کلمات آن‌ها برای هر مقاله است. با استفاده از این الگوریتم برای داده‌های تردد ماشین‌ها نیز می‌توانیم دقیقا به چنین توزیعی برسیم.

یکی از متغییرهای بسیار مهم در این بخش، cluster_center است، که تعداد کلاسترهای نهایی را مشخص می‌کند . با تغییر این متغییر می‌توانیم تعابیر متفاوتی از داده داشته باشیم. اما یکی از واضح‌ترین نتیجه‌ها برای $\text{cluster\_center} = 3$ به دست می‌آید که آن را در شکل‌های زیر می‌بینید:



۲.۳ تحلیل نتایج

مهم‌ترین نکته‌ی این سه تصویر، روند تغییر ترددها در هر روز است. در دسته‌ی اول، تردد در ساعات اولیه روز افزایش می‌یابد، در هنگام ظهر کاهش پیدا می‌کند، سپس دوباره در شب افزایش پیدا می‌کند. در دسته‌ی دوم، تردد در صبح کم است، اما کم کم افزایش می‌یابد و در شب به اوج خود می‌رسد. دسته‌ی سوم روندی دقیقا عکس دسته‌ی دوم دارد، و بیشترین تردد را در صبح دارند و سپس کاهش می‌یابد.

سه دسته‌ی بالا را می‌توانیم به این صورت تفسیر کنیم. دسته‌ی اول نقاط پر تردد شهر هستند، که هم در روز و هم در شب تردد بالایی دارند. این نقاط احتمال مکان‌هایی وسط شهر هستند که تمام طول روز تردد دارند (البته طبیعتا تردد در ظهر کاهش می‌یابد). دسته‌ی دوم، احتمالا مکان‌های دیدنی و تفریحی و یا بازارهای شبانه هستند که در طول روز تردد زیادی ندارند (به دلیل این که مردم مشغول کار و مدرسه و ... هستند). دسته‌ی سوم نیز احتمالا مکان‌هایی هستند که در طول روز تردد بالایی دارند، مانند مکان‌های اداری، مسیر مدارس و ادارات، یا دوربین‌های نزدیک به مثلا نانوایی‌ها.

یک نکته‌ی بسیار جالب دیگری که در این تصاویر دیده می‌شود، این است که که تردد در روزهای جمعه، شنبه و یک شنبه به طرز جالبی پایین است. با چک کردن این ۳ روز روی تقویم، فهمیدم که این روزها تعطیل رسمی بوده‌اند (قیام ۱۵ خرداد و شهادت امام جعفر صادق (ع)). بسیار جالب است که این کاهش ترددها، فقط در دسته‌ی سوم رخ داده‌است، که دقیقا با شهود ما همخوانی دارد، که دسته‌ی سوم مکان‌هایی مانند مدارس و ادارات هستند. همچنین، الگوی کاهشی تردد در این سه روز از بین رفته است که باز هم مطابق با الگوی پیدا شده است.

این روش دسته‌بندی کردن دوربین‌ها می‌تواند فواید زیادی از جمله هدایت ترافیک، مکان درست بیلبوردهای تبلیغاتی، مکان مناسب برای بعضی از فروشگاه‌های خاص و ... داشته باشد.

Pixie ۴

۱.۴ توضیحات کلی

در این بخش، با استفاده از الگوریتم‌های خانواده‌ی Page Rank و در واقع با الگوریتمی شبیه به الگوریتم Pixie داده ها را تحلیل می‌کنیم. برای این کار، پس از تمیزکردن داده‌های اولیه، تنها ماشین‌های پرتردد را نگه می‌داریم.

برای استفاده از الگوریتم، یک گراف دو بخشی در نظر می‌گیریم، که یک سمت آن ماشین‌ها قرار دارند و سمت دیگر آن دوربین‌ها. سپس بین هر ماشین، و هر دوربینی که آن ماشین از آن رد شده، یک یال در نظر می‌گیریم که وزن آن برابر با تعداد بارهایی است که دوربین آن ماشین را دیده است. سپس تمام یال‌ها را به صورت یک لیست مجاورت نگه می‌داریم (که به کمک آن می‌توانیم یال‌های خروجی از هر راس را به دست آوریم).

حال الگوریتم Pixie را پیاده‌سازی می‌کنیم. این الگوریتم هم می‌تواند از یک دوربین شروع شود، و هم از یک ماشین. بدون کاستن از کلیت تنها یک سمت آن را توضیح می‌دهیم، اما کد هر دو سمت در فایل jupyter موجود است.

با شروع از دوربین اولیه، لیست تمامی ماشین‌های دیده‌شده در آن دوربین را انتخاب می‌کنیم و به صورت تصادفی به یکی از ماشین‌ها می‌رویم. سپس بازهم لیست تمام دوربین‌هایی که آن ماشین در آن‌ها دیده شده را در نظر می‌گیریم و تصادفی به یکی از آن دوربین‌ها می‌رویم. برای هر دوربین یک counter نگه می‌داریم و هربار که از یک دوربین (روی گراف) عبور می‌کنیم، آن را یک واحد افزایش می‌دهیم. پس از طی شدن تعداد مرحله‌ای مشخص، لیست counter ها را خروجی می‌دهیم.

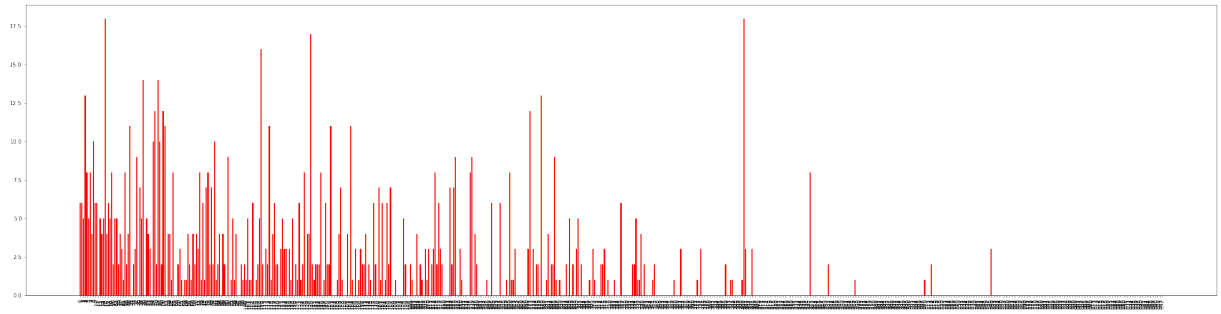
چند نکته‌ی حائز اهمیت در این الگوریتم وجود دارد. اول آن که لیست راس‌های سمت مقابل، واقعا به صورت «لیست» نگه داشته شده است. یعنی ممکن است عضوی تکراری داشته باشد. این کار از عمد صورت گرفته است، تا در واقع شانس انتخاب راس‌هایی با یال با وزن بیشتری وصل شده‌اند بیشتر شود. در واقع انتخاب راس سمت مقابل با این که تصادفی است، اما از یک توزیع احتمال پیروی می‌کند که احتمال انتخاب یال‌های با وزن بالا را متناسب با وزن آن افزایش می‌دهد. نکته‌ی بعدی نیز این است که به احتمال ثابتی، هر بار به راس اولیه برمی‌گردیم که این احتمال در کد من برابر با 0.2 بوده است.

۲.۴ تحلیل نتایج

حال که راجب نحوه‌ی پیاده‌سازی و کلیات این ایده صحبت کردیم، می‌توانیم بیشتر در مورد کاربردهای آن صحبت کنیم.

اولین کاربردی که این الگوریتم دارد این است که می‌تواند دوربین‌هایی که از نظر فیزیکی به هم نزدیکند را پیدا کند. زیرا اگر دو دوربین به هم نزدیک باشند، در صورتی که یک ماشین در یکی از آن‌ها دیده شود، به احتمال بیشتری در دیگری نیز دیده می‌شود. دقت کنید که عواملی مثل نوع دوربین در اینجا تاثیر گذارند. یعنی شباهت دوربین‌ها در حالت کلی تنها وابسته به فاصله‌ی فیزیکی نیست. در ادامه به این مورد بیشتر خواهم پرداخت.

در حالت کلی، اگر یک دوربین به عنوان ورودی داده شود، می‌توانیم شبیه‌ترین دوربین‌ها به آن را با این روش به دست آوریم. برای مثال یک نمونه از خروجی الگوریتم به صورت زیر است:



همانطور که می‌بینید، به کمک این اطلاعات می‌توانیم دوربین‌های نزدیک را بیابیم. حال اگر یک گراف رسم کنیم، که هر راس آن یک دوربین باشد، و هر دوربین را به چند دوربین نزدیک خود وصل کنیم، می‌توانیم نقشه‌ای حدودی از شهر داشته باشیم (حتی اگر موقعیت جغرافیایی دوربین‌ها را نداشته باشیم). «چند» دوربین نزدیک نیز می‌تواند عددی بین ۲ تا ۶ باشد (برای نتایج منطقی تر). دقت کنید که با این روش، اگر تنها لیست مکان دوربین‌ها را داشته باشیم، و اسم دوربین‌ها کد گذاری شده باشد، احتمال دارد بتوانیم موقعیت دقیق هرکدام از دوربین‌ها را روی نقشه بیابیم! پس این روش در حالت کلی می‌تواند به ما دوربین‌های مجاور را بدهد اما کارهای دیگری نیز مانند آنچه گفته شد می‌توان انجام داد.

نکته‌ی مهمی که حائز اهمیت است، این است که «نزدیکی» دوربین‌ها با تحلیل من لزوما نمی‌تواند به معنای نزدیکی فیزیکی باشد. زیرا اگر دو دوربین هردو تخلفات یکسانی را ثبت کنند، احتمال بیشتری وجود دارد که ماشین‌های یکسان را ثبت کنند. یعنی «نزدیکی» در این تحلیل می‌تواند نزدیک بودن نوع دوربین‌ها باشد. یک مثال از این اتفاق را می‌توانید در کد من ببینید. در بخشی از کد، نزدیک‌ترین دوربین‌ها به دوربین کوئری خود را در نظر گرفتم و نوع آن‌ها را چاپ کردم. نوع دوربین کوئری من ۸۱ بود، یعنی دوربین محدوده‌ی طرح ترافیک. و همانطوری که در نتایج دیده می‌شود، ۱۰ دوربین نزدیک همگی از نوع ۸۱ یا ۲۸۳ هستند که نوع ۲۸۳ نیز به معنی دوربین طرح زوج و فرد است (که در واقع یک دوربین محدوده طرح ترافیک است).

با استاده از مطلب بالا، می‌توانید ببینید که ما بدون دانستن اطلاعتی مانند نوع دوربین، می‌توانیم دوربین‌های با نوع مشابه و نزدیک به هم را پیدا کنیم. یعنی اگر نوع دوربین‌ها به ما داده نشده بود احتمالا می‌توانستیم دوربین‌ها را بر اساس شباهت‌هایی که از این الگوریتم به دست می‌آید دسته بندی کنیم.

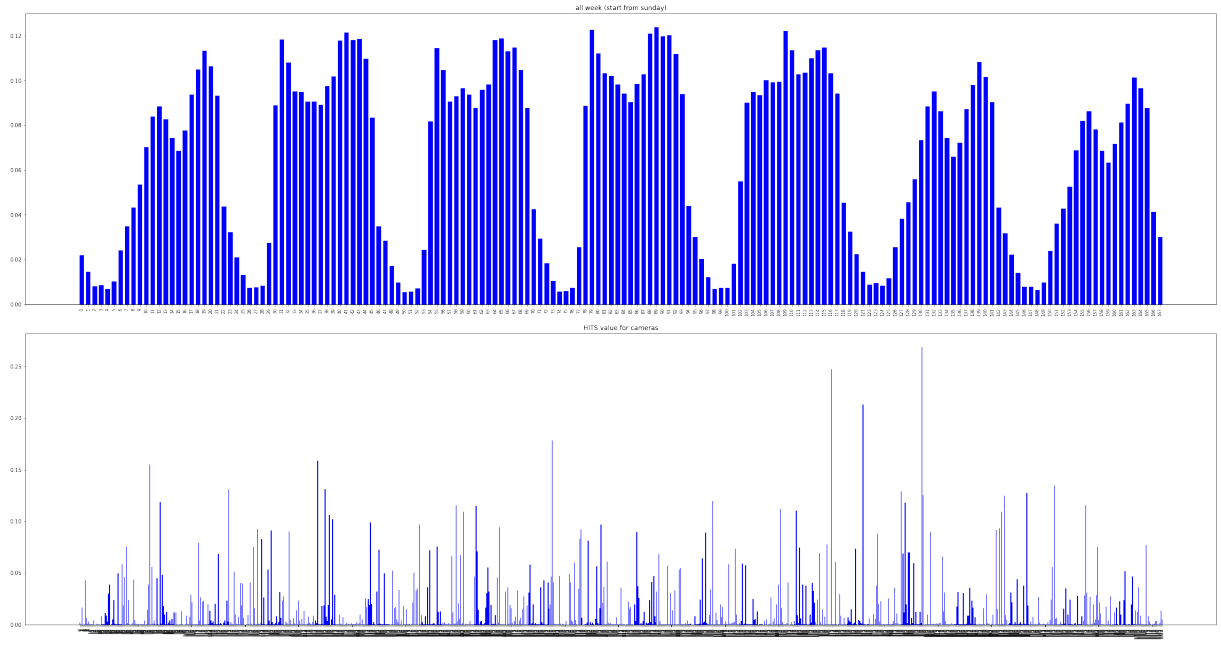
نکته‌ی دیگری که تحلیل بالا مشخص می‌کند این است که اگر دوربین‌های نوع‌های مختلف را حذف کنیم و فقط آن‌هایی را نگه داریم که به صورت کلی ثبت می‌کنند (نه به دلیل اتفاق خاصی، مانند نوع ماشین یا یک تخلف خاص) آنگاه می‌توانیم ایده‌ای که در بالا مطرح شد را بهتر پیاده سازی کنیم، و دیگر دوربین‌های نزدیک واقعا به معنی دوربین‌هایی هستند که از نظر فیزیکی به هم نزدیکند و به این صورت می‌توانیم نقشه‌ی گراف شهر را بکشیم.

حال به الگوریتم از سمت دیگر نگاه کنید. این الگوریتم می‌تواند ماشین‌های «نزدیک» به هم را تشخیص دهد. دقیقا مانند بالا، نزدیک بودن ماشین‌ها می‌تواند معیارهای معیارهای متفاوتی داشته باشد. برای مثال، اگر یک ماشین خاص را شناسایی کنیم که تعداد زیادی تخلف سرعت غیرمجاز دارد، با کمک این الگوریتم می‌توانیم ماشین‌هایی را بیابیم که الگوی حرکتی مشابهی با این ماشین دارند و در واقع ماشین‌هایی را بیابیم که رانندگی پرخطر دارند. تمام تحلیل‌های بالا را می‌توان برای این حالت نیز به کار گرفت و می‌توانیم ماشین‌هایی هم «نوع» را به دست بیاوریم.

HITS ۵

۱.۵ توضیحات کلی

کدهای ابتدایی این قسمت بسیار شبیه به حالت‌های قبلی هستند. در این الگوریتم، دوربین‌ها به عنوان hub و زمان‌ها به عنوان auhtority در نظر گرفته‌شده‌اند. سپس با استفاده از تجزیه‌ی SVD الگوریتم HITS را پیاده‌سازی کردم. نتایج نهایی به صورت زیر بودند:



۲.۵ تحلیل نتایج

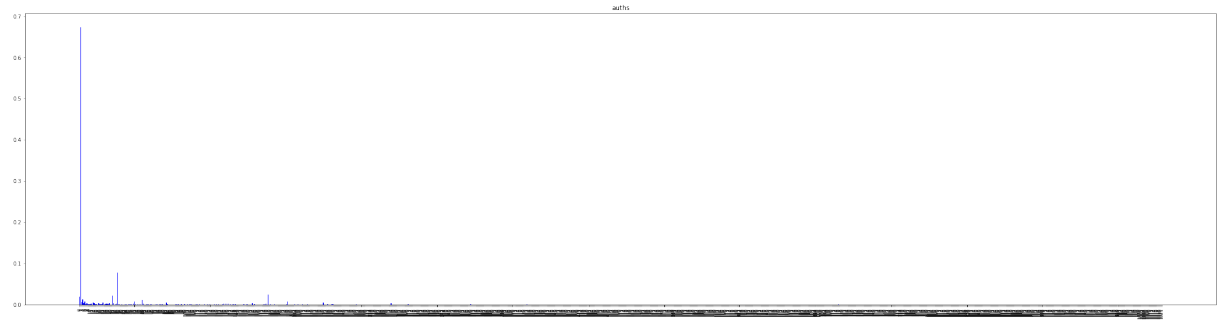
مقادیر این دو نمودار، به ما زمان‌ها یا دوربین‌هایی را نشان می‌دهند که به احتمال بیشتری ثبت می‌کنند. در واقع این نمودار دوربین‌ها می‌تواند به ما بگوید که کدام دوربین‌ها در ساعات پرتردد احتمال ثبت بیشتری دارند. اما این اطلاعات خیلی سودآور نیست. اما در مورد زمان‌ها، نمودارها اطلاعاتی دارند که با شهود ما سازگار است. همانطوری که می‌بینید، روزهای دوشنبه تا چهارشنبه حالت U شکل دارند، که این یعنی دو پیک ترافیکی یکی در صبح و یکی در شب داریم. اما این الگو در روز پنج شنبه، احتمالا به دلیل تعطیلی مدارس از بین می‌رود. سپس در روزهای جمعه و شنبه و یک‌شنبه، همانطوری که می‌بینید پیک شب قوی تر بوده است، که دلیل آن تعطیلی این سه روز است، که مردم بیشتر در شب تردد دارند (تعطیلی مدارس و ادارات).

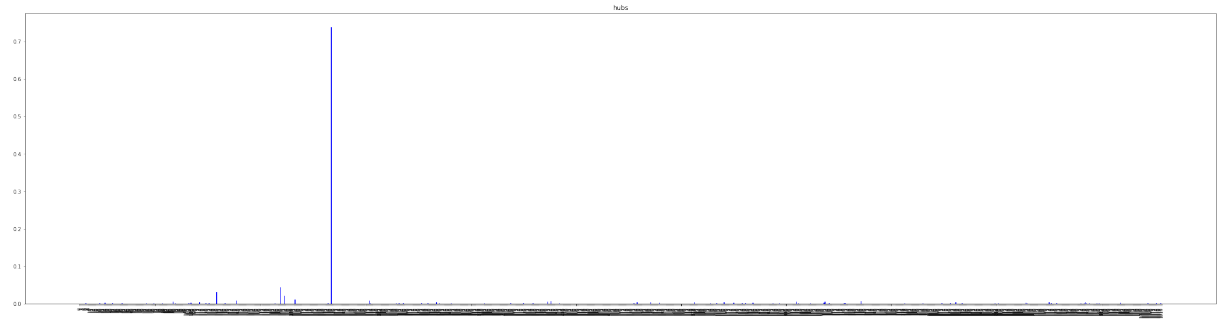
Important Cameras ۶

۱.۶ توضیحات کلی

در این قسمت، سعی کردم از الگوریتم HITS استفاده‌ی بهتری کنم. این بار، راس‌های من تنها شامل دوربین‌ها می‌شوند. کافی است برای هر ماشین در هر روز، دوربین‌هایی که پشت هم می‌آیند را یک یال گراف در نظر بگیرم. دقت کنید که این یال‌ها جهت دارند. حال اگر از الگوریتم HITS استفاده کنیم، می‌توانیم مهم‌ترین مکان‌هایی را پیدا کنیم، که به مکان‌های مهم دیگری مسیر دارند (بهترین hub ها و authority ها).

برای انجام دادن این کار، از تابع lead در sql استفاده می‌کنیم. برای هر ردیف دوربین ردیف بعدی را به df اضافه می‌کنیم. این کار باید به ازای هر ماشین و هر روز جداگانه انجام شود. همچنین باید دقت کنید که داده باید قبل از این قسمت روی زمان مرتب شود (lead یک سینتکس sql است. در صورتی که این قسمت گنگ است، کافی است توضیحات lead را بخوانید). نتیجه به صورت زیر شد:





۲.۶ تحلیل نتایج

همانطوری که می‌بینید، دو دوربین با اختلاف بهترین hub و authority شدند. لیست این دوربین‌ها به ترتیب در فایل jupyter آمده است (البته دقت کنید که دوربین‌ها اندیس گذاری شده‌اند و اندیس آن‌ها را برنگرداندم). نکته‌ی جالبی که وجود دارد این است که با این که دوربین‌ها به ترتیب کل تکرار شدن شماره گذاری شده‌اند، اما بهترین authority ۴۰ مین دوربین پرتکرار ما بوده است (یعنی با این که ۳۹ دوربین از آن پر تکرار تر بوده‌اند، اما بهترین authority شده است).

CF_ALS ۷

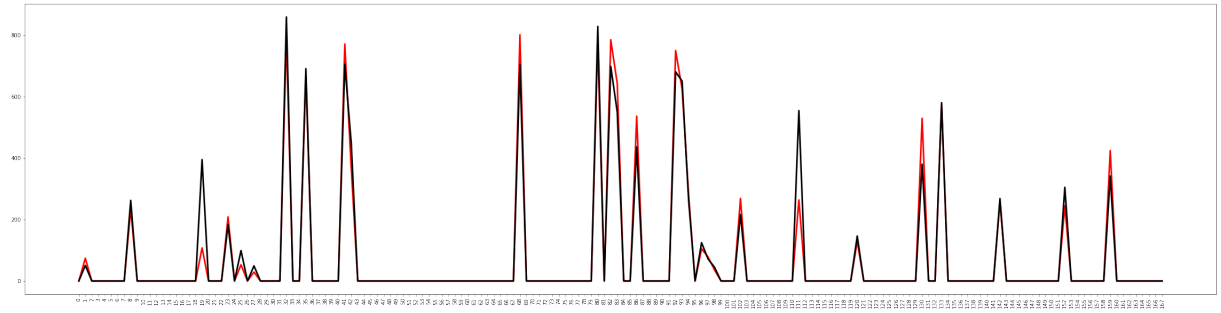
۱.۷ توضیحات کلی

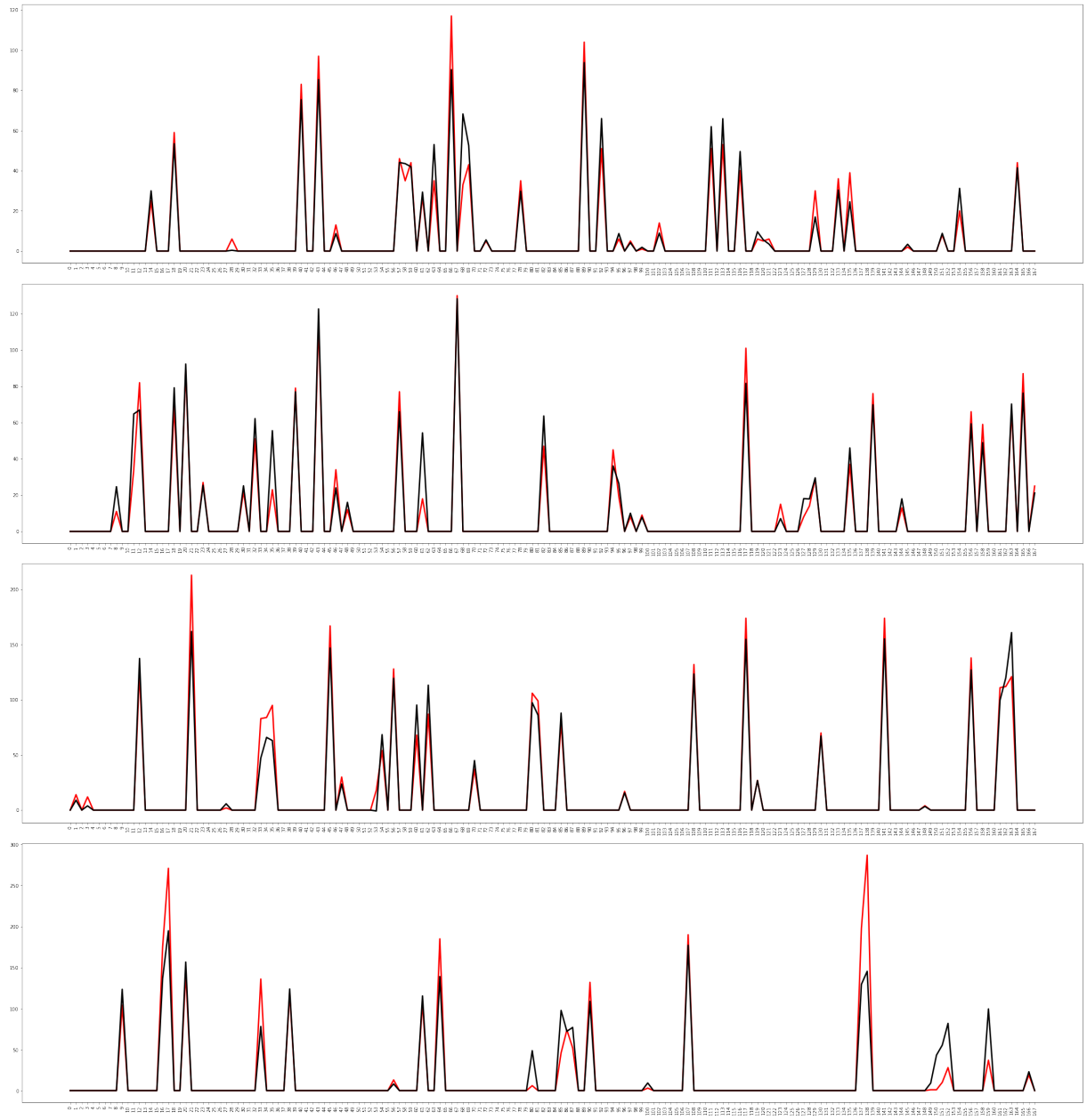
در این قسمت، روشی مانند روش Collaborative Filtering به کمک الگوریتم ALS پیاده‌سازی شده است. ابتدا برای هر دوربین و هر ساعت از هفته، تعداد ترددها را به دست آوردم. سپس داده را به دو قسمت آموزش و آزمون تقسیم کردم. با کمک داده‌های آموزش مدل ALS را train کردم. سپس با کمک داده‌های آزمون دقت مدل خود را سنجیدم.

مدل ما قابلیت این را دارد که به عنوان یک Recommender System عمل کند. در واقع می‌تواند برای هر دوربین و هر ساعت از هفته، تخمین بزند که چند ماشین در دوربین ثبت می‌شوند. کاربردهای آن می‌تواند به این صورت باشد که به ما کمک کند تصمیم بگیریم آیا یک دوربین در یک زمان خاص نیاز است روشن باشد یا خیر (مثلا ممکن است بتوانیم بسیاری از دوربین‌ها را فقط برای چند روز خاص به کار بگیریم). ممکن است یک دوربین در چند روز ترافیک شدیدی را ثبت کند، در این صورت می‌توانیم الگوریتم‌هایی پیش‌نهاد دهیم که به کمک این داده، برای روزهای بعد مسیرهای جایگزینی به راننده‌ها پیشنهاد دهد.

۲.۷ نتایج

برای آزمایش مدل خود، علاوه بر توابع آماده که در کد دیده می‌شود، چند دوربین به دلخواه انتخاب کردم و داده‌های آنان که در قسمت تست قرار داشت را در نظر گرفتم. دقت کنید که مدل ما هیچ وقت این داده‌ها را ندیده است. سپس مقادیر واقعی این داده‌ها و مقادیر پیش‌بینی شده را با هم بر روی نمودار نشان دادم. یعنی برای برخی ساعات از روز حدس زدم که یک دوربین خاص، احتمالا چند ماشین ثبت می‌کند و آن را با مقدار واقعی مقایسه کردم. خطوط قرمز برای داده‌های واقعی و خطوط سیاه مقادیر پیش‌بینی شده هستند. همانطوری که می‌بینید دقت مدل تا حد قابل قبولی بالا است.





۸ SVD

در این قسمت، ماتریسی ساختم که هر درایه‌ی آن نشان دهنده‌ی تعداد ثبت شدن یک ماشین خاص در یک دوربین خاص است. پس از تجزیه‌ی SVD مقادیر منفرد را بررسی کردم تا بتوانم در آن الگویی پیدا کنم. اما نتوانستم الگوی مشخصی پیدا کنم و این مقادیر را به مفهوم‌های خاصی پیوند دهم. برای تصویر کردن دوربین‌ها و ماشین‌ها روی فضای concept هم تلاش کردم اما نتیجه قابل فهم نبود. در نهایت بااین که کد این قسمت را کامل پیاده‌سازی کردم اما به نتیجه‌ی مطلوبی از نظر مفهومی نرسیدم.

۹ بقیه ایده‌ها

ایده‌ی دیگری که داشتم اما فرصت نکردم سراغ آن بروم، این بود که با استفاده از زمان‌های ثبت شده‌ی یک ماشین خاص در یک دوربین، پارک بودن آن ماشین را تشخیص دهم. برای مثال یک معیار ساده می‌توانست این باشد که در کمتر از یک ساعت، یک ماشین بیش از ۵ بار دریک دوربین دیده شود. با چنین روش‌هایی می‌شود نقاط قابل پارک کردن را پیدا کرد، و یا حتی دسته‌ای از ماشین‌ها که بیشتر پارک می‌کنند را جدا کرد. حتی می‌توان قصد ماشین‌های مختلف را نیز تشخیص داد (عبوری یا در حال پارک). همچنین از الگوریتم‌های streaming به دلیل کمبود وقت نتوانستم استفاده کنم. در قسمت‌هایی

راجب پیدا کردن نقشه‌ی شهر صحبت کردم، که می‌توانستم به دنبال کتاب‌خانه‌های مختلفی بگردم تا شاید بتوانم نتیجه‌ای قابل نمایش به دست آوردم و با نقشه‌ی واقعی نقاطی که دوربین‌ها قرار دارد تطبیق دهم.