دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده برق

پایان‌نامه کارشناسی‌ارشد
گرایش الکترونیک دیجیتال

آنالیز اطلاعات دریافتی از خودروها برای بررسی رفتار راننده و موقعیت مکانی خودروها در ابر محاسباتی

نگارش
حسین غلامی

استاد راهنما
دکتر معتمدی

پاییز ۹۹

چکیده

رانندگی خطرناک یکی از مهم‌ترین عوامل تصادفات خودرویی است. اگر رانندگان بدانند که رفتار آن‌ها در طول رانندگی ثبت می‌شـــود، رانندگی ایمن‌تری از خود ارائه خواهند داد. همچنین اگر بتوان با آگاهی از نوع رفتارهای رانندگی اشـخاص، سـیسـتمی را پیاده کرد که مشـوق افراد برای رانندگی ایمن باشـد، این اتفاق به کاهش تصادفات رانندگی منجر خواهد شد. در این پروژه هدف پیاده‌سازی روشی برای تشخیص رویدادهای رانندگی (ازجمله؛ گردش‌به‌راســـت و چپ خطرناک، تعویض لاین خطرناک به راســـت و چپ، ترمزگیری و شتاب‌گیری خطرناک) با کمک اطلاعات سنسورها است و پیاده سازی سیستمی که توانایی دریافت و ذخیره‌سازی اطلاعات رانندگان را در یک ابرمحاسباتی داشته باشد و همچنین روشی که میزان ایمن بودن رانندگی افراد را محاســبه کند. به‌صـورت دقیق‌تر، با اسـتفاده از یک روش جدید پنجره‌بندی روی ســـری‌های زمانی، یک درخت تصـــمیم‌گیری بر روی شـــباهت پنجره‌ها (با بهره‌گیری از الگوریتم Fast-DTW[1])، برای تشخیص رویدادهای رانندگی آموزش داده شد. در ادامه برای دریافت و ذخیره‌سازی اطلاعات از RabbitMQ به‌عنوان مبادله‌گر پیام و از Redis به‌عنوان پایگاه‌داده موقت بهره‌برداری شـــد. در پایان نیز برای امتیازدهی و مقایسـه رانندگی افراد، از یک روش آماری اسـتفاده شـد که میزان احتمال خطرناک بودن رانندگی آن‌ها را به‌عنوان معیاری برای مقایسـه معرفی می‌کند. با کمک نتایج ارائه‌شـده در این پژوهش می‌توان یک طرح بیمه‌ای پیاده‌سازی نمود تا رانندگانی که رانندگی ایمن‌تری را از خود نشان می‌دهنـد، حق بیمـه کمتری پرداخـت کننـد، ازاین‌رو می‌توان این طرح را بـه‌عنوان مشـــوقی برای بهبود فرهنگ رانندگی در نظر گرفت.

واژه‌های کلیدی:

اینترنت اشیاء، الگوریتم‌های هوش مصنوعی نمایه‌سازی رفتار راننده، فرهنگ رانندگی.

[1] Dynamic time warping

فهرست مطالب

صفحه

فهرست اشکال

صفحه

فهرست جداول

صفحه

۱

فصل اول

مقدمه

۱-۱ پیشگفتار و اهمیت موضوع

یکی از مهم‌ترین دلایل تصادفات جاده‌ای و مرگ‌ومیر، رانندگی خطرناک است. آمار نشان می‌دهد که دلیل ۷۰ درصد از تصادفات داخل شهری و ۹۰ درصد تصادفات جاده‌ای ناشی از خطای انسانی است [۱]، در [۲] که نتایج تصادفات داخل شهر تهران را بررسی می‌کند، ۸۶ درصد علت تصادفات را بی‌توجهی به قوانین راهنمایی رانندگی اعلام می‌کند و دلیل اصلی را فقر فرهنگ رانندگی می‌داند.

مطالعات نشان می‌دهد راننده‌ها اگر نسبت به ذخیره شدن نوع رفتار رانندگی خودآگاه باشند، رانندگی ایمن‌تری از خود ارائه خواهند داد [۳]، که منجر به کاهش آمار تصادفات و ایمن‌تر شدن خیلبان‌ها و جاده‌ها می‌شود، که هدف نهایی ماست. حال سوالی که مطرح می‌شود این است که چرا یک راننده باید اجازه دهد که رفتار و نوع رانندگی‌اش ذخیره و ما به آن اطلاعات دسترسی داشته باشیم. برای پاسخ به این سوال باید یک مشوق برای جلب توجه رانندگان به این امر ایجاد کنیم. برای مثال، به‌عنوان مشوق می‌توان، طرحی را در قالب "راننده‌ای که رانندگی ایمن‌تری از خود ارائه می‌دهد، می‌تواند حق بیمه کمتری پرداخت کند." تعریف کرد. این طرح اولا باعث می‌شود که رانندگان اجازه دسترسی به نوع رفتارشان را در اختیار شرکت بیمه‌ای قرار دهند تا حق بیمه کمتری پرداخت کنند، دوما ازآنجایی‌که رانندگان سعی می‌کنند رانندگی ایمن‌تری از خود ارائه دهند باعث می‌شود تا هدف اصلی ما که رانندگی ایمن است محقق شود. در مقابل، چون رانندگی خطرناک باعث می‌شود که افراد پول بیشتری را پرداخت کنند، رانندگی خطرناک به‌عنوان یک عامل بازدارنده عمل و تحقق هدف اصلی که رانندگی ایمن است را ساده‌تر می‌کند. همچنین می‌توان از منابع مالی حاصل‌شده از افرادی که رانندگی خطرناک دارند، برای ایمن‌سازی خودروها یا جاده‌ها و ... استفاده کرد، که سود ناشی از این طرح را دوچندان می‌کند. چالش اصلی این طرح هزینه تجهیزاتی است که بر روی خودرو نصب خواهند شد. این دستگاه‌ها می‌تواند نقش کلیدی در همه‌گیری این طرح داشته باشند و باید سعی شود، تا جایی که امکان دارد، قیمت تمام‌شده این دستگاه‌ها مناسب باشد. چالش بعدی تعمیر، نگهداری و طول عمر این دستگاه است.

۱-۲ طرح مسئله

بهتر است مسئله‌ای را در همین ابتدا با بررسی یکی از اصلی‌ترین چالش‌های پروژه شروع کنیم تا درک بهتری از مسئله پیش رو داشته باشیم. این چالش را با یک مثال ساده و با اعداد بررسی می‌کنیم؛ در تهران بر اساس [٤] بیش از ۴ میلیون خودرو درحرکت‌اند. با فرض نمونه‌گیری از حسگرها با فرکانس ۵۰ هرتز برای تشخیص رویدادها و آماده کردن بسته‌های ۱ کیلوبایتی، حجم ارسال اطلاعات در ثانیه،۲۰۰ گیگابایت است که برای ذخیره‌سازی، پردازش و ... حجم زیادی است. پس واضح است که با یک مسئله دادگان انبوه مواجه هستیم. اگرچه می‌توان این حجم از اطلاعات را کنترل نمود ولی همواره به دنبال راهکارهای ساده‌تری هستیم تا جایگزین شیوه فوق شود. در این پژوهش از شیوه دیگری استفاده شد، به این شکل که روند تشخیص رویدادهای خطرناک، به داخل هر خودرو منتقل شود و با راهکاری بتوان در زمان واقعی، رویدادها را تشخیص داد و به‌جای همه‌ی اطلاعات تنها رخدادهای خطرناک را به ابر منتقل کرد. پس با توجه به توضیح فوق، برای تحقق هدف اصلی پروژه که پیاده‌سازی روشی برای تشخیص رویدادهای رانندگی و تجمیع و بررسی اطلاعات رانندگان در ابر محاسباتی است، پروژه به سه بخش متفاوت تقسیم شد که به شرح زیر است:

- طراحی مدلی که از اطلاعات سنسوری رخدادهای خطرناک رانندگی را تشخیص دهد.
- پیاده‌سازی محیطی در ابر که عمل دریافت و ذخیره‌سازی اطلاعات را به انجام رساند.
- طراحی یک مدل آماری که رانندگی افراد را مقایسه کند و به رفتارهای رانندگان امتیاز دهد.

اطلاعات سنسوری به‌کاررفته در این پژوهش اطلاعات حسگرهای شتاب‌سنج و ژیروسکوپ است که هر دو حسگرهای ارزان‌قیمتی محسوب شده و احتمال خراب شدن آن‌ها بسیار کم است که با بهره‌گیری از اطلاعات آن‌ها توانستیم رخدادهای خطرناک رانندگی ازجمله؛ گردش به چپ و راست شدید، ترمز و شتابگیری شدید، تعویض لاین به چپ و راست شدید را با دقت قابل قبولی تشخیص دهیم. این فرایند با آموزش یک مدل درخت تصمیم‌گیری[۱] که بردارهای ویژگی آن از یک پنجره‌بندی اتفاقی و الگوریتم Fast-DTW بدست آمده، بر روی یک مجموعه دادگان که برچسب‌های رخدادهای فوق را داشتند، انجام

[۱] id۳ with prune

شـــد. در ادامه با کمک Docker، با به‌کارگیری برنامه‌های RabbitMQ، Redis و برنامه‌ای میانی که این دو نرم‌افزار را به هم متصل کند سیستمی یکپارچه بر بستر ابری طراحی شد تا به‌عنوان ابر محاسباتی وظیفه دریافت و ذخیره‌ســـازی اطلاعات را انجام دهد. در پایان نیز با کمک مجموعه دادگانی دیگر که حجم اطلاعات بیشـــتری در اختیار ما قرار می‌داد یک مدل مبتنی بر توزیع‌های آماری طراحی شـــد تا با کمک آن بتوانیم نحوه عملکرد رانندگان را تشخیص دهیم و آن‌ها را مورد ارزیابی قرار دهیم.

۱-۳ ساختار پایان نامه

در این رساله، که در هفت فصل تدوین شده، در ابتدای هر فصل ابتدا مروری بر مفاهیم کلی، الگوریتم‌ها و نرم‌افزارهای به‌کاررفته، انجام شده و در ادامه جزئیات در بخش‌ها و طراحی آن را شرح داده خواهد شد. در فصل دوم به مروری بر کارهای انجام شده در این زمینه اشاره میشود و س در فصل سوم به طراحی مدلی که رخدادهای خطرناک را تشـــخیص می‌دهد، پرداخته شـــده، در فصـــل چهارم به پیاده‌ســـازی ابرمحاسـباتی که وظیفه دریافت و ذخیره‌سـازی را دارد، اشـاره شـده، در فصـل پنجم نحوه امتیاز دهی به رانندگان توضیح داده شده، در فصل ششم، عملکرد هر یک از بخش‌ها به طور کامل بررسی می‌شود. در فصـــل هفتم نیز به نتیجه‌گیری و پیشـــنهادهایی که در ادامه راه این پروژه می‌تواند انجام شـــود، پرداخته می‌شود.

۲ فصل دوم

مروری بر کارهای پیشین

مروری بر کارهای پیشین

به‌طورکلی تشـــخیص رفتار راننده به معنی فرایندی خودکار برای جمع‌آوری اطلاعات راننده (ســـرعت، مکان، شـتابگیری، ترمزگیری، نوع هدایت کردن و ...) و اعمال یک عملیات محاسـباتی است تا امتیازی به میزان ایمن بودن رانندگی فرد، تخصـــیص یابد [۵]. اطلاعات از خودروها ممکن اســـت به طرق متفاوتی جمع‌آوری شـوند، برای مثال اطلاعات می‌توانند با کمک سـنسـورهای تلفن همراه هوشـمند یا با کمک سـیسـتم‌های قابل توسـعه یا با کمک دوربین‌های متصـل به خودرو جمع‌آوری شـود. همچنین روش‌های متفاوتی برای انجام عملیات محاســباتی برای تشـــخیص رفتار راننده وجود دارد که در ادامه به بررســی تعدادی از پژوهش‌ها و پروژه‌هایی که در این زمینه‌ها منتشرشده‌اند می‌پردازیم.

برنامه کاربردی Nericell که در [۶] معرفی شـــد، یک برنامه تلفن همراه اســـت که وضـــعیت جاده‌ها و ترافیک آن‌ها را بررسی می‌کند. این برنامه از سنسور شتاب سنج تلفن‌های هوشمند برای تشخیص چاله‌ها و رخدادهای ترمزگیری بهره می‌گیرد. همچنین از میکروفن تلفن همراه برای تشـــخیص صـــدای بوق استفاده می‌کند، این برنامه با کمک GPS نسبت به موقعیت مکانی و سرعت خودرو آگاه می‌شود. شیوه‌ی تشخیص ترمز و دست‌اندازهای خیابان با رسیدن میزان اندازه حسگر شتاب سنج به سطح آستانه‌ای از پیش تعیین‌شده در پنجره‌های زمانی مشخصی، است. نتایج و دقت استخراج رویداد در این برنامه با معیار تشـخیص غلط و تسـت درسـت (FP) و تشـخیص غلط و تسـت غلط (FN) به ترتیب برای رخدادهای ترمزگیری: ,۲٪FP:۲۲,۲٪-FN:٤,٤٪ و برای تشـــخیص دســـت‌انداز و چاله دقت : ٪FP:٥٪-FN:۲۳٪ در سرعت‌های زیر ۲۵ کیلومتر بر ساعت انجام شد.

در [۷] یک برنامه کاربردی اندروید بلادرنگ، برای تشـــخیص رانندگی خطرناک و تشـــخیص رانندگی در حالت مستی ارائه شد. این برنامه به‌طور مستقیم از اطلاعات حسگر شتاب سنج و گردش‌سنج[۱]، حرکات دورانی غیرعادی خودرو و مسـئله عدم توانایی ثابت نگه‌داشـتن سـرعت را (که هر دو از علائم رانندگی در حالت مستی هستند) را تشخیص می‌دهد. سپس با حل معادلاتی با کمک اطلاعات دو حسگر فوق، بردار شـــتاب طولی و عرضـــی خودرو را محاســـبه می‌کند. برای تشـــخیص رویدادهای غیرعادی دورانی، در

[۱] Orientation Sensor : این حسگر یک حسگر فیزیکی نیست بلکه خروجی یک فیلتر کالمن است که از اطلاعات حسگرهای شتاب سنج و ژیروسکوپ و در مواقعی مگنومتر(تشخیص میدان مغناطیسی زمین) تشکیل شده و خروجی این حسگرها پارامترهای roll,pitch,yaw است که با توجه به نحوه قرارگیری تلفن همراه مشخص می‌شود.

پنجره‌های پنج‌ثانیه‌ای میزان ماکزیمم و مینیمم سیگنال را محاسبه کرده و درصورتی‌که از حد آستانه‌ای مشخص عبور کند، هشدار علائم دورانی فعال می‌شود و درصورتی‌که بردار طولی سرعت در طی زمانی مشخص مثبت یا منفی باقی بماند، هشدار عدم توانایی ثابت نگه‌داشتن سرعت فعال می‌شود، نتایج عملی این پژوهش به‌صورت روبه‌رو است: FN: ۰٪-FP: ۰٫٤۹٪ برای تشخص حرکات دورانی غیرعادی و FN: ۰٪-FP: ۲٫۹٪, برای تشخیص توانایی کنترل ثابت نگه‌داشتن سرعت.

برنامه کاربردی WreckWatch که در [۸] معرفی شد، یک برنامه اندرویدی تحت شبکه برای تشخیص تصادفات رانندگی است. برنامه سمت کاربر، تصادفات را تشخیص و اطلاعات مربوطه را تهیه‌کرده و به سرویس‌دهنده ارسال می‌کند. این برنامه اطلاعات حسگر شتاب‌سنج، GPS و میکروفن را برای تشخیص تصادفات، مورداستفاده قرار می‌دهد. این برنامه ۱۱ حالت مختلف وضعیت تلفن همراه را برای تشخیص تصادف بررسی می‌کند. پارامترهایی که برای تشخیص تصادف در نظر گرفته می‌شوند، تغییر ناگهانی شتاب در یک‌جهت، بررسی صدای بلند و سرعت خودرو هستند. یکی از سناریوهای تشخیص تصادف می‌تواند وقوع تغییر ناگهانی شتاب، ایجاد صدای بلند و توقف خودرو باشد. نتایج آزمایشات عملی نشان می‌دهد که درصورتی‌که تلفن همراه از روی سطح داشبورد به زمین بیافتد، نتیجه مشابه تصادف و در بعضی از تصادفات به علت سایر شرایط محیطی میکروفن افزایش صدا را تشخیص نداده و تصادف تشخیص داده نشده است ولی در اکثر موارد توانسته به‌خوبی عمل کند.

در [۹] یک برنامه تلفن همراه هوشمند ارائه‌شده که میزان مصرف سوخت را محاسبه می‌کند و از این طریق تأثیرات رفتار راننده بر مصرف سوخت را تحلیل می‌کند. همچنین این برنامه توانایی راهنمایی (هشدار تعویض دنده یا سرعت زیاد) در حین رانندگی را داراست تا راننده عملکرد بهتری در جهت کاهش مصرف سوخت از خود ارائه دهد. در این برنامه بجای اینکه از حسگرهای تلفن همراه هوشمند استفاده شود، دستگاهی به پورت OBD-II خودرو متصل می‌شود که اطلاعات ECU خودرو (نظیر سرعت، شتاب، دور موتور و...) را توسط بلوتوث به تلفن همراه منتقل کند. تلفن همراه وظیفه ذخیره و نگهداری این اطلاعات را بر عهده دارد. در ادامه این اطلاعات به کمک یک تبدیل خطی و دو سیستم فازی میزان مصرف سوخت را اعلام می‌کردند و همه پردازش به‌صورت بلادرنگ انجام می‌شود.

در [۱۰]، یک برنامه بر روی تلفن همراه ارائه‌شده که توانایی دسته‌بندی رفتار راننده به ایمن و پرخطر را دارا است. این برنامه توانایی تشخیص رویداد چرخش، ترمزگیری و شتابگیری خطرناک را دارد و برای

این کار از حسگرهای شتاب‌سنج و ژیروسکوپ، مگنومتر استفاده می‌کند. به این شکل که ابتدا نقاطی را روی سیگنال را مشخص کرده، سپس با مقایسه الگوهایی از پیش تعیین‌شده میزان شباهت سیگنال‌ها را با کمک DTW محاسبه می‌کند و در پایان با کمک یک دسته‌بندی کننده بیزین باینری، به رویدادها برچسب خطرناک و ایمن می‌زند، دقت تشخیص رخداد خطرناک از غیر خطرناک در آن ۸۷ درصد است.

در[۱۱] یک برنامه کاربردی اندروید برای تلفن همراه نوشته‌شده، که با کمک حسگرهای شتاب‌سنج و GPS، وضعیت خودرو (جابه‌جایی و سرعت) ، الگوهای رانندگی (شتابگیری و ترمزگیری)، وضعیت جاده(خاکی، اسفالت ، دست‌انداز، چاله) را شناسایی می‌کند. تشخیص این رخدادها با محاسبه اندازه مقادیر حسگر با آستانه‌ای مشخص، تکرار و تعداد رخداد ایجادشده و زمان میان این رویدادها انجام می‌شود. به‌عنوان‌مثال، برای شتاب گیری و ترمزگیری بردار شتاب در راستای y نباید از میزان ۰.۳g بیشتر شود. نتایج عملی این پژوهش دقت دسته‌بندی جاده‌ها را برای تشخیص نقاط غیرعادی در جاده‌های خاکی؛ دست‌انداز: ٪۸۱.۵ ، چاله:٪۷۵ ، در جاده‌های آسفالت: ؛ دست‌انداز: ٪۹۱۵ ، چاله:٪۸۹.۴ اعلام کرده است.

Sensfleet که در [۱۲] معرفی‌شده، یک برنامه کاربردی تلفن همراه هوشمند برای نمره‌دهی به رفتار راننده است که توانایی تشخیص رخدادهای خطرناک دارا است. این برنامه از اطلاعات حسگرهای شتاب‌سنج، مگنومتر، گرانش‌سنج و GPS استفاده کرده که با کمک یک سیستم فازی در صورتی که حسگر ها به آستانه ای مشخص برسند، رخدادهای خطرناک رانندگی ازجمله سرعت زیاد، چرخش خطرناک، شتابگیری و ترمز خطرناک که در یک سفر رخ می‌دهد را شناسایی می‌کند. این برنامه در انتهای هر سفر، عددی بین ۱۰۰-۰ به هر راننده می‌دهد که میزان ایمن بودن رانندگی وی را بر اساس مسافتی که شخص راننده طی کرده، مشخص می‌کند.

در [۱۳] با طول های مختلفی، اطلاعات حسگرهای شتابسنج، ژیروسکوپ و مگنومتر را پنجره‌بندی کرده و تنها با محاسبه متغیرهای آماری هر پنجره (واریانس، میانگین هر پنجره، و ضریبی از میانگین پنجره قبل) بردار ویژگی را استخراج می‌نماید و با بکارگیری طبقه بندی کننده‌های مختلف (شبکه عصبی، SVM و شبکه‌های بیزین) سعی می‌کند رخدادهای خطرناک رانندگی را تشخیص دهد. بیشترین دقت مجموعه از بهره‌گیری یک شبکه های عصبی، با ۲ لایه مخفی و پنجره‌ای با طول ۲۰۰ بدست می‌آید. همچنین در این پژوهش اهمیت سنسورهای بکار رفته را بررسی کرده که بر اساس نتایج به این نکته

دست یافته که میزان اهمیت حسگرهای شتاب‌سنج و ژیروسکوپ نسبت به مگنومتر بیشتر بوده و مدل بر اساس اطلاعات این دو حسگر رخدادها را تشخیص می‌دهند.

در [۱٤] شروع به پنجره بندی بر سیگنال های شتابسنج، ژیروسکوپ و مگنومتر کرده و بجای اینکه بردارهای ویژگی هر پنجره محاسبه شود، اطلاعات خام حسگر ها به شبکه های عصبی بازگشتی داده شده و سعی شده که با کمک شبکه های SimpleRNN،GRU و LSTM رخدادهای خطرناک رانندگی تشخیص داده شوند. که نتیجه کلی هر یک از شبکه ها به ترتیب ۹۴، ۷۰ ، ۹۵ درصد است. که شبکه عصبی LSTM با ۱۰ نرون بیشترین دقت را داشته است.

در [۱۵] نرم‌افزاری برای تلفن همراه طراحی‌شده است که اطلاعات شتاب‌سنج، مگنومتر و سرعت متوسط خودرو که توسط GPS استخراج‌شده را پنجره‌بندی کرده، سپس نویز آن را حذف نموده و در ادامه با گرفتن تبدیل ویولت[۱]، با استفاده از بردار ویژگی استخراج‌شده، یک شبکه عصبی با دقت ۷۹ درصد از رویدادهای خطرناک، نیمه خطرناک و ایمن آموزش داده است.

در [۱٦] برنامه‌ای برای تشخیص رخدادهای رانندگی به‌صورت ایمن، خطرناک طراحی‌شده است، پنجره‌های زمانی در این طراحی ۳.۵ ثانیه و به‌صورت ثلبت است. این برنامه با استفاده از اطلاعات کامپیوتر مرکزی خودرو، دو مدل یادگیری عمیق با به‌کارگیری یک شبکه LSTM (با تعداد سلول و لایه‌های متفاوت) و یک شبکه چهار لایه‌ای از CNN یک‌بعدی و یک شبکه fully connected سه لایه‌ای در ادامه آن، سعی بر تشخیص رویدادهای خطرناک کرده است، میزان دقت Fl-score برای بهترین مدل شبکه LSTM (پس بهینه‌سازی تعداد و لایه) به عدد ٪۸۲ رسیده و میزان دقت Fl-score شبکه حاوی CNN یک‌بعدی به عدد ٪۹۳ رسیده است. در این پژوهش، رویدادها به‌طور جزئی استخراج نمی‌شوند و تنها رویدادها به‌صورت خطرناک و ایمن بررسی می‌شوند.

در [۱۷] با استفاده از حسگرهای شتاب‌سنج و ژیروسکوپ، رخدادهای ترمزگیری، شتابگیری و تغییرلاین خطرناک با به‌کارگیری شبکه عصبی عمیقی تشخیص داده‌شده است. این شبکه که از دولایه CNN یک‌بعدی و یک شبکه fully connected تشکیل‌شده، رویدادها را به‌طور متوسط با دقت

[۱] Wavelet transform

Fl-score، ۷۱/۵درصد شناسایی می‌کند. زمان محاسبه هر رویداد توسط این روش ۴.۱۵ ثانیه است؛ که به این معنی است که در زمان واقعی نمی‌توان از این الگوریتم استفاده نمود.

بر اساس پژوهش‌هایی که در پیش تعریف شد؛ به‌طورکلی ۳ رویکرد در مواجهه با مسئله تشخیص و نمره‌دهی رفتار رانندگان وجود دارد: ۱- رویکرد رسیدن حسگر به آستانه‌ای مشخص، به صورتی که حسگر به حد آستانه‌ای مشخص نزدیک شود، رخدادها تشخیص دهد، به‌عنوان‌مثال تشخیص رویداد سرعت غیرمجاز که اگر سرعت به حد آستانه‌ای برسد، رویداد تشخیص داده خواهد شد. ۲- استخراج بردار ویژگی و به‌کارگیری الگوریتم‌های یادگیری ماشین، با استفاده از اطلاعات حسگرها و یک بردار ویژگی با روش‌های مختلف (تبدیل ویولت،DTW و...) استخراج شود و با کمک الگوریتم‌های یادگیری ماشین (مدل‌های بیزین، درخت تصمیم،...) سعی بر تحلیل رفتار راننده شود. ۳- استفاده از روش‌های یادگیری عمیق، که با کمک اطلاعات سنسوری موجود در یک شبکه‌های عصبی طراحی و آموزش داده شود به‌نحوی‌که بتواند به تشخیص رخدادهای خطرناک یا امتیازدهی به رانندگی بپردازد.

مزیت روش اول این است که به‌سادگی و با حداقل هزینه پردازشی می‌تواند به‌صورت بلادرنگ پیاده‌سازی شود ولی با این روش نمی‌توان رویدادی‌های پیچیده‌ای را با دقت بالا تشخیص داد. روش دوم شیوه‌ای میانه است به این معنا که می‌توان به تشخیص مدل‌های پیچیده‌تر پرداخت و بسته به نوع پردازنده مورداستفاده، می‌توان آن را به‌صورت بلادرنگ پیاده‌سازی کرد؛ اما مزیت روش سوم توانایی تشخیص هرگونه رویداد پیچیده‌ای است که بتوانیم برچسب اطلاعات آن را استخراج کرده و شبکه عصبی متناسب به آن را آموزش داد، ولی عیب اصلی آن بزرگ بودن مدل ها و حجم پردازش زیاد برای پیاده‌سازی در سخت افزارهای قابل توسعه است که پیاده‌سازی آن را در زمان واقعی با چالش مواجه می‌کند.

۳ فصل سوم

مدل پیشنهادی در تشخیص رخدادهای خطرناک

تشخیص رخدادهای خطرناک

در این بخش ابتدا مروری بر الگوریتم‌ها انجام شـــده و ســـپس به بررســـی اطلاعات و مجموعه دادگان مورداستفاده پرداخته می‌شود تا درک بهتری از اطلاعاتی که با آن کار می‌کنیم، داشته باشیم. در ادامه به بررسـی شـیوه آموزش مدل پرداخته می‌شـود. در آخر از بین مدل‌های آموزش داده‌شـده، بهترین مدل که بیشترین دقت توانایی پیاده‌سازی در عمل را دارد انتخاب می‌شود.

۳-۱ مروری بر الگوریتم‌های FastDTW

۳-۱-۱ الگوریتم DTW

DTW[۱] یک تکنیک برای یافتن میزان شـباهت دو سـری زمانی و یا همسـویی بهینه بین دو سـری زمانی اســت. یک ســری زمانی ممکن اســت با بزرگ یا کوچک کردن در امتداد محور زمانی خود، به‌صــورت غیرخطی «تاب[۲]» داده شـود. سـپس می‌توان از این تاب دادن میان دو سـری زمانی، برای یافتن مناطق متناظر بین این دو سـری یا تعیین شـباهت این دو سـری اسـتفاده کرد. از DTW اغلب برای تشـخیص اینکه آیا دو شـکل موج گفتار، یک عبارت گفتاری یکسـان را نشـان می‌دهند یا خیر، اسـتفاده می‌شـود. در شـکل موج گفتار، مدت‌زمان هر صـدای گفتاری و فاصـله بین صـداها، مجاز اسـت که تغییر کند، اما شـکل موج کل گفتار باید مشـابه باشـد. علاوه بر تشـخیص گفتار، DTW در بسـیاری از رشـته‌های دیگر ازجمله داده‌کاوی، تشـخیص حرکت و رباتیک نیز مفید شـناخته‌شـده اسـت.DTW معمولاً در داده‌کاوی به‌عنوان اندازه‌گیری فاصله بین سری‌های زمانی استفاده می‌شود. نمونه‌ای از نحوه «تاب دادن» یک سری زمانی به سری زمانی دیگر در شکل ۳–۱ نشان داده شده است.

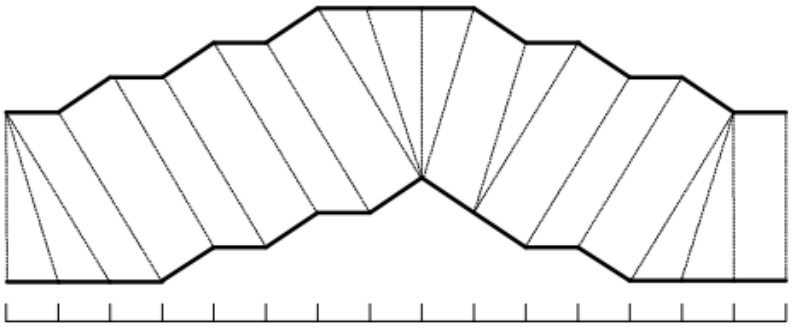
در شـکل ۳–۱ هر خط عمودی یک نقطه را در یک سـری زمانی به نقطه مشـابه آن در سـری زمانی دیگر متصـل می‌کند. خطوط درواقع مقادیر مشـابهی در محور y دارند اما از هم جداشـده‌اند؛ بنابراین می‌توان خطوط عمودی بین آن‌ها را با سهولت بیشتری مشاهده کرد. اگر هر دو سری زمانی در این شکل یکسان

---

۱ Dynamic time warping

۲ Warped

باشند، همه خطوط مستقیماً خطوطی عمودی خواهند بود زیرا برای «ردیف کردن[1]» این دو سری زمانی هیچ‌گونه تاب نخواهیم داشت. فاصله مسیر تاب، اندازه‌گیری تفاوت بین این دو سری زمانی است که بعد از انحراف آن‌ها باهم جمع می‌شود که از جمع فواصل بین هر جفت از نقاط متصل به خطوط عمودی در شکل ۳–۱ اندازه‌گیری می‌شود؛ بنابراین، دو سری زمانی که به‌جز تاب موضعی که نسبت به محور زمان یکسان هستند، فاصله DTW صفر دارند. علی‌رغم کاربردی بودن الگوریتم DTW، دارای پیچیدگی زمانی و مکانی $O(N^2)$ است که مفید بودن آن را به سری‌های زمانی کوچک حاوی بیش از چند هزار نقطه داده محدود می‌کند.



شکل ۳–۱ مفهوم تاب دادن میان دو سری زمانی [۱۷]

اندازه‌گیری فاصله بین سری‌های زمانی برای تعیین شباهت بین آن‌ها و برای طبقه‌بندی سری‌های زمانی مورداستفاده قرار می‌گیرد. فاصله اقلیدسی یک شیوه اندازه‌گیری فاصله است که می‌تواند مورداستفاده قرار گیرد. فاصله اقلیدسی بین دو سری زمانی، مجموع مربع فواصل از هر نقطه n ام تا نقطه n ام در سری زمانی دیگر است. عیب اصلی استفاده از فاصله اقلیدسی برای داده‌های سری زمانی این است که نتایج آن بسیار غیرشهودی است. اگر دو سری زمانی یکسان باشند، اما یکی در طول محور زمان کمی جابجا شود، فاصله اقلیدسی ممکن است آن‌ها را بسیار متفاوت از یکدیگر بداند. برای مقابله با این محدودیت اندازه‌گیری فاصله بین سری‌های زمانی با نادیده گرفتن تغییرات و شیفت‌های زمانی DTW در [۱۸] معرفی شد، به‌اختصار آن را توضیح داده و در ادامه به بررسی Fast-DTW می‌پردازیم.

با فرض داشتن دو سری زمانی X و Y به طول‌های $|X|$ و $|Y|$ داریم :

[1] Line up

$$Y = y_1, y_2, \dots, y_j, \dots, y_{|Y|} \qquad X = x_1, x_2, \dots, x_i, \dots, x_{|X|}$$ رابطه (۳-۱)

طول مسیر تاب به این شکل تعریف می‌شود:

$$W = w_1, w_2, \dots, w_K \qquad \max(|X|, |Y|) \le K < |X| + |Y|$$ رابطه (۳-۲)

که در آن k تعداد طول مسیر تاب و k اُمین عنصر مسیر تاب $w_k = (i, j)$ تعریف می‌شود. به صورتی که در آن i ، درایه سری زمانی X و j درایه سری زمانی Y باشد.

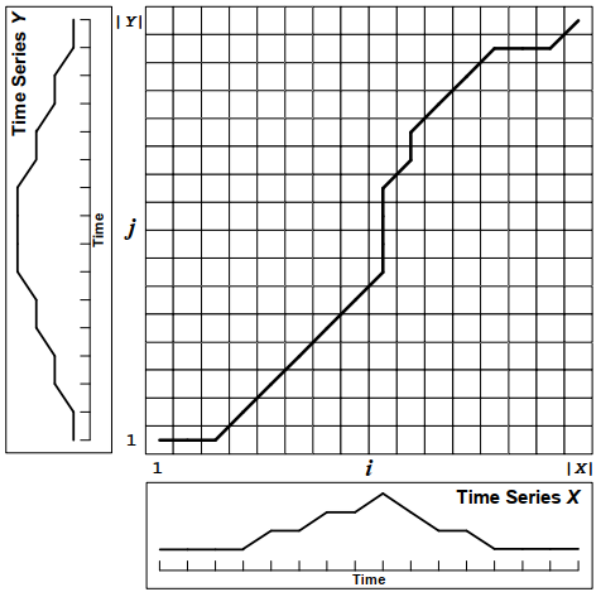
مسیر تابش باید از (۱،۱) شروع‌شده و با (|X|,|Y|) تمام شود تا این اطمینان را بدهد که از هر عنصر دو سری زمانی در مسیر تاب استفاده می‌شود. در مسیر تاب نیز محدودیتی وجود دارد که i و j را مجبور می‌کند تا به‌طور یکنواخت در مسیر تاب افزایش یابند، به همین دلیل خطوط نمایانگر مسیر تاب در شکل ۳-۱ هم‌پوشانی ندارند. باید از هر درایه هر سری زمانی استفاده شود. به بیان ریاضی:

$$w_k = (i, j), w_{k+1} = (i', j') \; i \le i' \le i + 1, j \le j' \le j + 1$$ رابطه (۳-۳)

مسیر بهینه تاب، مسیری تابی است که طول مسیر تاب W، حداقل فاصله را طی کند:

$$\text{Dist}(W) = \sum_{k=1}^{k=K} \text{Dist}\left(w_{ki}, w_{kj}\right)$$ رابطه (۳-۴)

$\text{Dist}(W)$ فاصله‌ای است که می‌توان بین این دو سری زمانی (معیاری از شباهت) تعریف می‌شود.



شکل ۳-۲ -ماتریس هزینه، که در آن مسیر تاب بهینه مشخص‌شده است. [۱۷]

الگوریتم DTW، از یک رویکرد برنامه‌نویسی پویا برای یافتن این مسیر تاب بهینه استفاده می‌کند و به‌جای تلاش برای حل یک‌باره کل مسئله، مسئله را به بخش‌های کوچک‌تر (بخش‌هایی از سری زمانی) تقسیم کرده و سعی بر حل آن دارد تا جایی که مسئله نهایی حل شود. یک ماتریس هزینه D در ابعاد |X| در|Y| تعریف شد که میزان $D(i,j)$ آن مینیمم مسیر تاب است که توسط دو سری زمانی X و Y ساخته می‌شود. شکل ۳–۲ بیانگر یک نمونه از ماتریس هزینه است که مینیمم فاصله‌ی تاب بین دونقطه‌ی $D(۱,۱)$ و $D(|X|,|Y|)$ را نمایش می‌دهد.

برای یافتن مینیمم فاصله‌ی تاب، همه‌ی سلول‌های ماتریس هزینه باید حساب شوند. منطق استفاده از یک رویکرد برنامه‌نویسی پویا برای حل مسئله این است که چون $D(i,j)$ حداقل فاصله‌ی تاب از دوسری زمانی مشخص به طول i و j است، اگر حداقل فاصله‌های تاب برای همه قسمت‌ها کمی کوچک‌تر از آن سری زمانی که یک نقطه داده واحد با طول i و j فاصله‌دارند، مشخص باشد، سپس مقدار $D(i,j)$ حداقل فاصله تمام‌مسیرهای تاب ممکن برای سری‌های زمانی است که یک نقطه داده کوچک‌تر از i و j هستند، به‌علاوه فاصله بین دونقطه X_i و Y_j خواهد شد.

$$D(i,j) = \text{Dist}\,(i,j) + min[D(i-۱,j), D(i,j-۱)D(i-۱,j-۱)] \qquad \text{رابطه (۳–۵)}$$

برای مثال: دو سری زمانی X و Y به ترتیب به‌صورت $\{۰,۰,۶,۲,۶,۰,۰,۳,۲,۱\}$ و $\{۳,۳,۰,۰,۶,۲,۶,۰\}$ در اختیارداریم و می‌خواهیم میزان شباهت دو سیگنال را محاسبه کنیم، ابتدا باید ماتریس D را طبق رابطه ۳-۵ از نقطه $D(۱,۱)$محاسبه کنیم

جدول ۳–۱-نحوه محاسبه ماتریس هزینه در الگوریتم DTW

time series Y	cost matrix									
۳	۱۷+۳									
۳	۱۴+۳	۳+۱۴								
۰	۰+۱۴	۰+۱۴								
۰	۰+۱۴	۱۴+۱۴								
۶	۶+۸	۶+۸								
۲	۲+۶	۸+۶	۴+۰	۰+۰						
۶	۶	۶	۰+۰	۴+۰						
۰	۰	۰	۶	۲+۶	۶+۸	۰+۱۴	۰+۱۴	۱۴+۳	۱۷+۲	۱۸+۲
time series X	۰	۰	۶	۲	۶	۰	۰	۳	۲	۱

جدول ۳-۲-محاسبه مسیر تاب و میزان شباهت دو سیگنال توسط DTW

time series Y	cost matrix									
۳	۲۰	۲۰	۱۸	۶	۸	۶	۶	۰	۱	۳
۳	۱۷	۱۷	۱۵	۵	۷	۳	۳	۰	۴	۶
۰	۱۴	۱۴	۱۲	۴	۸	۰	۰	۳	۵	۶
۰	۱۴	۲۸	۶	۲	۶	۰	۰	۳	۵	۶
۶	۱۴	۱۴	۰	۴	۰	۶	۱۲	۳	۵	۱۰
۲	۸	۱۴	۴	۰	۴	۶	۸	۱	۵	۶
۶	۶	۶	۰	۴	۴	۱۰	۱۶	۳	۵	۱۰
۰	۰	۰	۶	۸	۱۴	۱۴	۱۴	۱۷	۱۹	۲۰
time series X	۰	۰	۶	۲	۶	۰	۰	۳	۲	۱

همان‌طور که در جدول ۳-۲ مشاهده می‌کنید، پس از تکمیل ماتریس D، برای محاسبه میزان شباهت از نقطه (|X|,|Y|) شروع کرده و بر روی کم‌ترین میزان، حرکت می‌کنیم تا به نقطه‌ی (۱,۱) برسیم حال مجموع اعداد مسیر طی شده را با یکدیگر جمع می‌کنیم؛ که به عدد چهار می‌رسیم، همان‌طور که مشاهده می‌شود اکثر مسیر بهینه تاب عدد صفر را نمایش می‌دهد که به این معنی است که دو سری زمانی X و Y با یکدیگر نسبتاً شباهت بالایی دارند، همان‌طور که می‌دانستیم این دو سری زمانی انتقال‌یافته یکدیگر بودند که در نقاط اندکی با یکدیگر تفاوت داشتند.

نکته قابل‌توجه بعدی این است، هر چه از این مسیر بهینه دور می‌شویم این اعداد بزرگ‌تر می‌شوند درحالی‌که برای محاسبه شباهت دو سری زمانی به این اعداد نیاز نداریم، محاسبه‌ی این اعداد تنها باعث زمان‌بر شدن حل مسئله می‌شوند. همچنین در این الگوریتم برای یافتن مسیر بهینه می‌بایست همه‌ی درایه‌های ماتریس D را محاسبه می‌کردیم که شاید بتوان با ایده‌ای خلاقانه محاسبات را ساده و سریع‌تر کرد، خوشبختانه این کار پیش‌تر در [۱۷] انجام‌شده و الگوریتمی با عنوان FastDTW در آن ارائه شد. پیچیدگی محاسباتی در الگوریتم DTW در ردیف $O(N^2)$ و در الگوریتم FastDTW در ردیف $O(N)$ قرار دارد، در ادامه به شرح الگوریتم FastDTW می‌پردازیم.

۳-۱-۲ الگوریتم FastDTW

الگوریتم FastDTW از یک رویکرد چند سطحی با سه عملیات کلیدی استفاده می‌کند:

۱- درشـــت کردن[۱]: یک ســـری زمانی را به یک ســـری زمانی کوچک‌تر تبدیل می‌کند که منحنی یکسانی را با حداکثر دقت با نقاط داده کمتر نشان می‌دهد.

۲- پیش‌نمایش[۲]: یک مسـیر تاب بهینه را، در وضـوح کمتر پیدا می‌کند تا از آن به‌عنوان یک حدس اولیه برای مسیر تاب بهینه در وضوح[۳] بالاتر استفاده کند.

۳- اصلاح کردن[۴]: توسط همسایگی‌ها که در وضوح بالاتر محاسبه می‌شود، مسیر تاب پیش‌نمایش شده، در وضوح پایین، اصلاح می‌شوند.

درشـــت کردن، ســـایز (وضـــوح) ســـری زمانی‌ها را با میانگین‌گیری روی نقاط هم‌جوار کاهش می‌دهد. درنتیجه سـری زمانی جدید، از سـری زمانی قبلی کوچک‌تر اسـت. چندین بار این اتفاق می‌افتد تا سـری زمانی‌هایی با وضـوح متفاوتی ایجاد شـوند. **پیش‌نمایش** بر روی پایین‌ترین وضـوح، الگوریتم مسـیر تاب بهینه را محاسبه می‌کند. ازآنجایی‌که وضوح با ضریب دو افزایش می‌یابد، یک نقطه واحد، در مسیر تاب با وضوح پایین به حداقل چهار نقطه در وضوح بالاتر تبدیل می‌شود. این مسیر پیش‌بینی‌شده به‌عنوان یک روش ابتکاری در هنگام اصـلاح مسـیرتاب بهینه برای یافتن یک مسـیرتاب خورده با وضـوح بالاتر اسـتفاده می‌شود. **اصلاح کردن**، بهینه‌ترین مسیرتاب را در همسایگی مسیری (که در پیش‌نمایش معرفی شد) با کمک پارامتر شعاعی[۵] می‌یابد.

الگوریتم FastDTW ابتدا از درشـت سـازی برای ایجاد تمام وضـوح‌هایی که بررسـی می‌شـوند، اسـتفاده می‌کند. شکل ۳–۳ چهار وضوح متفاوت هنگام اجرای الگوریتم FastDTW را نشـان می‌دهد که پیش‌تر در سری زمانی X,Y که در شکل ۳–۱و شکل ۳–۲ استفاده‌شده است مشاهده می‌کنید.

---

۱ Coarsening

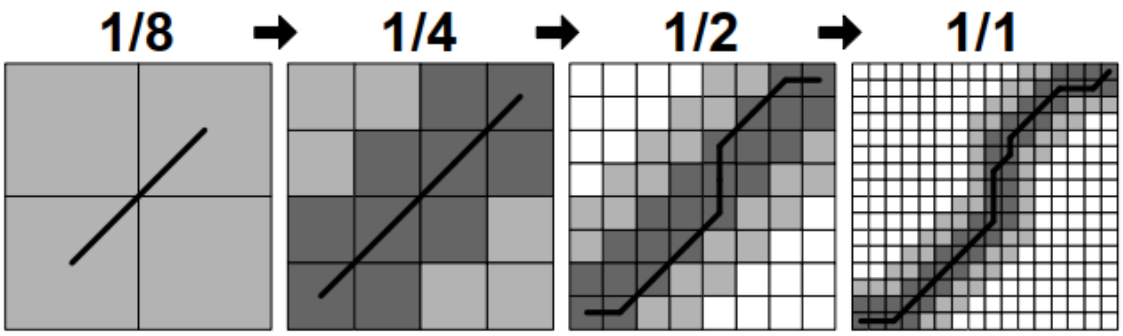
۲ Projection

۳ Resolution

۴ Refinement

۵ Radius

الگوریتم استاندارد DTW برای یافتن مسیر بهینه تاب برای سری زمانی با کمترین وضوح اجرا می‌شود. این مسیرتاب در سمت چپ شکل ۳–۳ که با وضوح کم است، نشان داده‌شده. بعدازاین که مسیر بهینه در این حالت محاسبه شد به حالت پیش‌نمایش در مرحله بعد ظاهر می‌شود. برای اصلاح کردن در این مرحله الگوریتم DTW به‌صورت محدودشده اعمال می‌شود. محدودیت هم شامل تنها محاسبه مسیر تابی است که پیش‌نمایش شده، به این شکل می‌توان مسیرتاب بهینه را از وضوح کمتر به‌وضوح بالاتر برد. اگرچه ممکن است که مسیرتاب بهینه اصلی در محدوده پیش‌نمایش وجود نداشته باشد، پس یک پارامتر دیگر تحت عنوان پارامتر شعاعی در نظر گرفته شد که نقاطی به مجموعه پیش‌نمایش برای بررسی مجدد اضافه کند. در شکل ۳–۳ پارامتر شعاعی ۱ در نظر گرفته شد. به‌این‌ترتیب هنگامی‌که محدوده پیش‌نمایش محاسبه شد، یک واحد از هر سمت به اطراف آن افزوده می‌شود (خاکستری کم‌رنگ).



شکل ۳–۳-چهار وضوح متفاوت در حین اجرای الگوریتم FastDTW [۱۷]

اگرچه مسیرتاب بهینه در شکل فوق وارد محدوده شعاعی نشد و محاسبه آن نیازی نبود ولی نکته قابل‌توجه در این الگوریتم به‌سادگی در شکل ۳–۳ قابل‌مشاهده است، در این الگوریتم تعداد خانه‌های ماتریس هزینه که ما حساب کردیم شامل ۴+۱۶+ ۴۴+۱۰۰ هستند که یعنی ۱۶۴ درایه محاسبه شد این درحالی‌که است که اگر الگوریتم DTW ساده را می‌خواستیم محاسبه کنیم باید ۲۵۶ خانه را بررسی می‌کردیم که به‌طور کامل کاهش حجم محاسبات را برای ما نشان می‌دهد.

۳-۲ جمع‌آوری و آنالیز اکتشافی اطلاعات

برای پیش برد این بخش، از مجموعه دادگان ارائه‌شده توسط [۱۹] استفاده شد. این مجموعه اطلاعات توسط تلفن همراه هوشمند جمع‌آوری گردید که شامل اطلاعات سنسوری شتاب‌سنج، شتاب‌سنج‌خطی، مگنومتر و ژیروسکوپ بود. جزئیات این اطلاعات به شرح زیر است:

۱- شامل ۴ مرتبه رانندگی به مدت‌زمان متوسط ۱۳ دقیقه است.

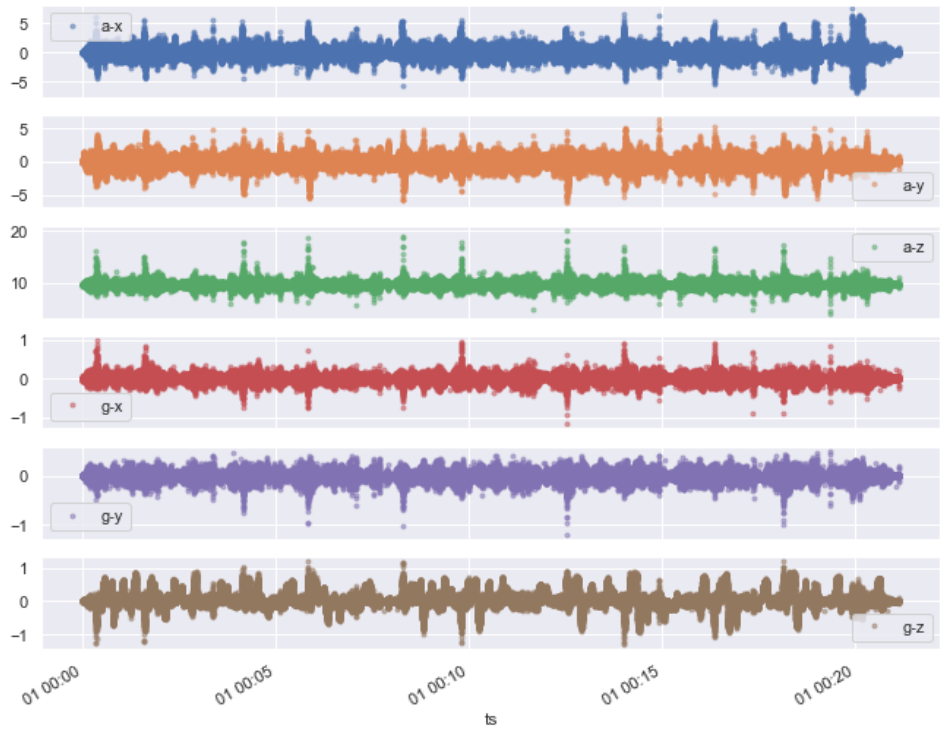
۲- شرکت‌کنندگان در این آزمایش، دو راننده باسابقه ۱۵ سال رانندگی بودند که رخدادهای خطرناک را ایجاد می‌کردند.

۳- فرکانس نمونه‌برداری از شتاب‌سنج و ژیروسکوپ ۵۰ هرتز است.

٤- آزمایش‌ها در آب‌وهوای آفتابی و در جاده‌هایی با آسفالت خشک انجام شد.

٥- تلفن همراه هوشمند مورداستفاده Motorola XT۱۰۵۸ با نسخه اندروید ۵.۱ انجام شد.

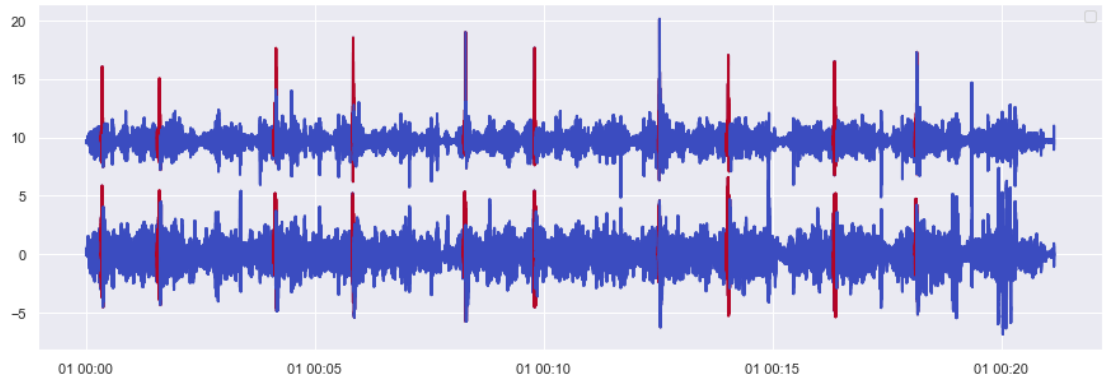
٦- تلفن هوشمند با استفاده از وسیله‌ای بر روی شیشه جلو اتومبیل Honda Civic ۲۰۱۱ ثابت شد تا هنگام جمع‌آوری داده‌های تلفن جابجا نشود.



شکل ۳-۴-نمونه‌ای از اطلاعات سنسوری یک راننده در طول ۲۰ دقیقه رانندگی

همان‌طور که در شکل ۳–۴ مشاهده می‌کنید، در این پژوهش ما تنها از اطلاعات حسگرهای شتاب‌سنج و ژیروسکوپ استفاده کردیم. برچسب‌گذاری رویدادها در فایلی جداگانه و به‌صورت زمان محور اعلام شد.

به این معنا که زمان شروع و اتمام رویداد ذخیره‌شده و در اختیار ما قرار می‌گیرد. برای مثال در شکل ۳–۵ که اطلاعات تمامی سنسورها به‌صورت تجمیع یافته نمایش داده شدند، لبه‌های قرمزرنگ، رویدادهای مربوط به ترمزگیری شدید هستند.



شکل ۳–۵- رویدادهای ترمزگیری شدید

توزیع طبقه‌بندی‌های این مجموعه دادگان را در جدول ۳-۳ مشاهده می‌کنید. همان‌طور که مشخص است درمجموع ۵۵ رخداد برای یک ساعت رانندگی وجود دارد که برای انجام فرایند طبقه‌بندی میزان مناسبی می‌تواند باشد و با توجه به گستردگی و تنوع از نوع رویدادها می‌توان فرایند تشخیص را به‌طور کامل انجام داد. حال به بررسی دقیق‌تر این اطلاعات می‌پردازیم:

جدول ۳–۳-توزیع طبقه‌بندی مجموعه دادگان

	تعداد	کوچک‌ترین	بزرگ‌ترین	میانگین	انحراف معیار
شتابگیری خطرناک	۱۲	۱۲۰	۲۴۵	۱۸۰	۴۲/۱۵
ترمزگیری خطرناک	۱۲	۹۵	۱۸۵	۱۱۹	۲۴/۷۲
چرخش به راست خطرناک	۱۱	۱۵۰	۲۲۵	۱۷۳	۲۵/۶
چرخش به چپ خطرناک	۱۱	۱۴۰	۲۲۵	۱۷۳	۲۴/۹
تعویض لاین به راست خطرناک	۵	۹۵	۱۲۵	۱۱۱	۱۱/۹
تعویض لاین به چپ خطرناک	۴	۸۰	۱۲۰	۱۰۵	۱۷/۷

می‌دانیم یکی از ساده‌ترین روش‌ها برای بررسی یک مجموعه دادگان مقایسه آن با توزیع نرمال است، همچنین می‌دانیم در توزیع نرمال ۹۵/۴۳ درصد اطلاعات در بازه [میانگین ± دو برابر انحراف معیار] قرار می‌گیرند، در جدول ۳-۴ محاسبه این امر را مشاهده می‌کنیم. می‌توان بررسی نمود که اطلاعات

نسبت به توزیع نرمال فشرده‌تر هستند که نشان از استاندارد بودن مجموعه دادگان مورد بررسی است و این امر کمک می‌کند که فرایند طبقه‌بندی را با دقت بهتری بتوانیم انجام دهیم.

جدول ۳-۴-مقایسه رویداد های مجموعه دادگان با توزیع نرمال

	کوچک‌ترین	μ-۲σ	بزرگ‌ترین	μ+۲σ
شتابگیری خطرناک	۱۲۰	۹۵/۷	۲۴۵	۲۶۴/۳
ترمزگیری خطرناک	۹۵	۶۹/۵۶	۱۸۵	۱۶۸/۴۴
چرخش به راست خطرناک	۱۵۰	۱۲۱/۸	۲۲۵	۲۲۴/۲
چرخش به چپ خطرناک	۱۴۰	۱۲۳/۲	۲۲۵	۲۲۲/۸
تعویض لاین به راست خطرناک	۹۵	۸۷/۲	۱۲۵	۱۳۴/۸
تعویض لاین به چپ خطرناک	۸۰	۶۹/۶	۱۲۰	۱۴۰/۴

۳-۳ طراحی مدل

اولین مسئله‌ای که در طراحی مدل برای تشخیص رویدادهای رانندگی با آن مواجه هستیم، نحوه نگاه کردن به سری‌های زمانی است. برای این کار سعی می‌کنیم از نحوه نگاه کردن انسان به سری‌های زمانی ایده بگیریم، برای مثال هنگامی‌که یک شخص که تحلیل سری زمانی‌هایی همچون نوار قلب را بلد نیست، می‌خواهیم بدانیم چگونه به این سیگنال نگاه می‌کند؛ اما یک پزشک که تحلیل این سری‌های‌زمانی را فراگرفته او چگونه به این نمودارها نگاه می‌کند. در حالت اول وقتی شخص، آموزش نیافته و نمی‌داند، باید به دنبال چه چیزی بگردد. ممکن است نگاهش را روی سیگنال عقب و جلو کند، بعضی جا را رد کند کمی به عقب‌تر برگردد و مجدد این سری‌های زمانی را بررسی کند. درحالی‌که یک شخص متخصص و آموزش‌دیده همچون پزشک، با نگاه کردن به‌صورت مستقیم دقیقا به دنبال یافتن الگوهایی در نوار قلبی می‌گردد و رویداد را تشخیص می‌دهد. ایده اولیه طراحی مدل پنجره‌بندی که در این پژوهش استفاده شد، از نحوه نگاه کردن به سری‌های زمانی و بررسی دقیق این مسئله برمی‌خیزد.

۳-۳-۱ پیش‌پردازش و پنجره‌بندی سری زمانی

همان‌طور که در مثال فوق دیدیم و همچنین در کارهای پیشین مشاهده کردیم، اکثر مدل‌ها شیوه پنجره‌بندی مستقیمی داشتند، به این معنا که پنجره‌ای که بر روی سری‌های زمانی حرکت می‌دهند را

به‌صورت مستقیم روی سیگنال حرکت می‌دادند، در این پژوهش ما شیوه جدیدی از پنجره‌بندی را ارائه کردیم که در ادامه به‌تفصیل شرح داده می‌شود.

وقتی پنجره را روی سیگنال حرکت می‌دهیم، به دنبال یافتن الگوهایی هستیم که پیش‌تر نسبت به آن‌ها آگاهیم، الگوهایی همچون شتابگیری خطرناک، ترمزگیری خطرناک و...، پس باید طول پنجره‌ها را بر اساس الگوهایی که قصد شناسایی آن را داریم طراحی کنیم. با توجه به جدول ۳-۳ می‌توانیم مشاهده کنیم که طول میانگین رویدادهای «شتاب گیری خطرناک»، «چرخش به سمت راست خطرناک» و «چرخش به سمت چپ خطرناک» و همچنین، طول میانگین پنجره‌های «تعویض لاین به سمت چپ خطرناک» و «تعویض لاین به سمت راست خطرناک» بسیار به یکدیگر نزدیک‌اند. ولی رخدادهای «ترمزگیری خطرناک» نسبت به سایرین متفاوت است. پس شاید بتوان رویدادهایی که طول نزدیک‌تری به یکدیگر دارند را با یک پنجره تشخیص داد؛ اما برای طول پنجره، چون می‌خواهیم همه‌ی رویدادها را تشخیص دهیم، باید طول بزرگ‌ترین رویداد را در نظر گرفت. پس تعداد و طول پنجره‌ها با استدلال فوق به دست خواهد آمد. در این طراحی سه پنجره با طول‌های ۲۲۶، برای رویدادهای «شتاب گیری خطرناک»، «چرخش به سمت راست خطرناک» و «چرخش به سمت چپ خطرناک»، ۱۲۶ برای رویدادهای «تعویض لاین به سمت چپ خطرناک» و «تعویض لاین به سمت راست خطرناک» و ۱۸۶ برای رویداد «ترمزگیری خطرناک» آماده شد.[۱] پس تا اینجا سه پنجره تعریف شد که بتوانند رخدادهای متفاوت را در خود جای دهند. این نحوه از حرکت دادن پنجره‌ها مدل ما را از overfit شدن دور کرده و حساسیت آن را نسبت به نقطه شروع سیگنال به حداقل می‌رساند.

برای پنجره‌بندی یک متغیر تصادفی گوسی M ، با میانگین x (مثلاً ۰.۱) و واریانس y (مثلاً ۰.۰۲۵) تعریف شد.[۲] که مسبب حرکت پنجره بر روی سری زمانی می‌شود. به این شکل که مرکز پنجره بر روی point قرارگرفته و میزان هر بار جلو رفتن پنجره توسط $Step_{forward}$ محاسبه می‌شود. در ابتدا point بر روی $Window_{lenght}/۲$ تنظیم‌شده و تا انتهای سیگنال حرکت می‌کند.

[۱] با توجه به فرکانس نمونه برداری از سنسور ها که ۵۰ هرتز می‌باشد ، طول زمانی پنجره ها به ترتیب ۴.۵۲ ثانیه ، ۲.۵۲ ثانیه و ۳.۷۲ ثانیه در نظرگرفته شد. همچنین دلیل زوج بودن طول پنجره ها به دلیل استفاده از الگوریتم FastDTW است که در مرحله درشت کردن، وضوح سیگنال را نصف می‌کند.

[۲] در ادامه در بخش بهینه سازی به تفصیل انتخاب x ، y تشریح می‌شود.

$$Step_{forward} = round(M * Window_{lenght})$$ رابطه (۳-۶)

$$point = point + Step_{forward}$$ رابطه ۳-۷)

برای مثال: M می‌تواند مجموعه اعداد $\{$ ۰.۱-، ۰، ۰.۱، ۰.۲، ۰.۳$\}$ باشد چراکه میانگین اعداد فوق ۰.۱ و واریانس آن‌ها ۰.۰۲۵ است، حال وسط پنجره به طول ۱۲۶ در ابتدا در نقطه‌ی ۶۳ قرارداد در ادامهM اعداد مجموعه را می‌گیرد،

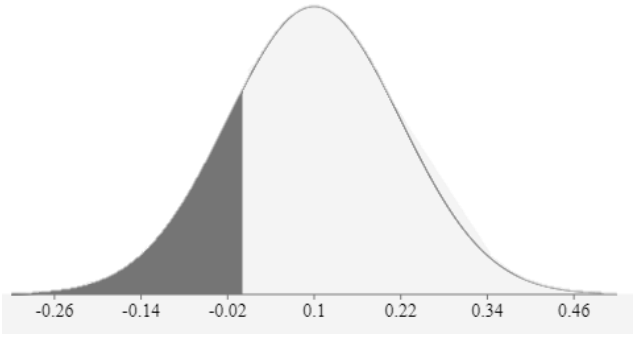
$$point = ۶۳\ , Step_{forward} = round(۰.۳ * ۱۲۶) = ۳۷, point^* = ۶۳ + ۳۷ = ۱۰۰$$

همان‌طور که بررسی شد، وسط پنجره به نقطه ۱۰۰ می‌رسد که به معنای جلو رفتن آن است. به همین شکل، هنگامی‌که M عددی منفی است، پنجره به عقب حرکت می‌کند.

$$point = ۱۰۰\ , Step_{forward} = round(-۰.۱ * ۱۲۶) = -۱۲, point^* = ۱۰۰ - ۱۲ = ۸۸$$

به این شکل با روش فوق می‌توانیم حالت‌های متفاوتی از سری زمانی را بررسی کنیم، نکته قابل‌توجه این است که عدد میانگین باید، بین صفر و یک باشد، چراکه درصورتی‌که از یک بزرگ‌تر باشد، لحظاتی از سری زمانی را هرگز بررسی نمی‌کنیم و همچنین درصورتی‌که کوچک‌تر از صفر باشد، پنجره به عقب حرکت خواهد کرد و همچنین میانگین هرچه به ۱ نزدیک شود تعداد نمونه‌های دیده‌شده کاهش و هرچه به صفر نزدیک شود، تعداد نمونه‌ها افزایش می‌یابد. اهمیت واریانس در میزان جابه‌جایی و پرش‌های سیگنال است به نحوی‌که، زیاد باشد میزان پرش به عقب و جلو زیاد شده و ممکن است بخش‌های از سری زمانی دیده نشود و درصورتی‌که کم باشد میزان حساسیت به نقطه شروع زیاد شده و دقت روی دادگان تست کاهش میابد.

برای مشخص کردن دقیق میزان واریانس، از پارامتر دیگری تحت عنوان «احتمال نگاه به رویدادهای پیشین» استفاده می‌کنیم، به این شکل که اگر عدد تصادفی به‌دست‌آمده منفی باشد، بدین معناست که به سمت عقب نگاه خواهیم کرد. برای مثال؛ متغیر تصادفی M را با میانگین ۰.۱ و انحراف معیار ۰.۱۲ در نظر بگیرید؛ همان‌طور که در شکل ۳-۶ توزیع این متغیر تصادفی را مشاهده می‌کنید، مساحت زیر نمودار به ازای اعداد کوچک‌تر از صفر برابر ۲۰ درصد مساحت کل است، به این معنا که احتمال نگاه کردن به عقب ۲۰ درصد می‌شود.



شکل ۳–۶-توزیع نرمال با میانگین ۰.۱ و انحراف معیار ۰.۱۲

پس از پنجره‌بندی فوق، خروجی این فرآیند سه ردیف اطلاعات بخش شده از سری‌های‌زمانی است که در یک دیکشنری پایتون قرار گرفت. در ادامه این پنجره‌ها را «پنجره‌های موقت» نام‌گذاری می‌کنیم. حالا می‌بایست برچسب هر یک از این پنجره‌ها را از فایلی که اطلاعات و برچسب‌ها در آن واقع‌شده، بر روی هر پنجره اعمال کرد، برای این کار یک تابع نوشته شد که بررسی کند آیا هرکدام از این پنجره‌ها با رویدادی که برچسب آن‌ها را داریم، تداخل دارند یا خیر. برای مثال:

در ردیف پنجره‌های موقت به طول ۱۸۶، که به دنبال رویداد «ترمزگیری خطرناک» هستیم، ۸۰۰۰ پنجره به طول ۱۸۶ داریم که بدون برچسب هستند و قصد داریم آن‌ها را برچسب بزنیم، با استفاده از تابعی که پیش‌تر ذکر شد، بررسی می‌کنیم که آیا در این پنجره رویداد ترمزگیری شدید وجود دارد یا خیر. در شکل ۳–۷ نمونه‌ای از این پنجره‌ها را مشاهده می‌کنیم.

('freada_agressiva',

ts	a-x	a-y	a-z	g-x	g-y	g-z
1970-01-01 00:02:18.175	0.556711	0.511286	0.724074	0.673125	0.414671	0.460563
1970-01-01 00:02:18.195	0.497366	0.473231	0.626607	0.683067	0.344634	0.429376
1970-01-01 00:02:18.214	0.513601	0.511313	0.608938	0.624408	0.448702	0.435434
1970-01-01 00:02:18.234	0.529479	0.571977	0.694869	0.548310	0.484986	0.390378
1970-01-01 00:02:18.254	0.542950	0.634868	0.667002	0.612514	0.401522	0.404699
...	...	...	...	...	...	...
1970-01-01 00:02:21.729	0.685795	0.834751	0.775077	0.631134	0.418013	0.398236
1970-01-01 00:02:21.749	0.584033	0.768531	0.692371	0.625630	0.395291	0.400530
1970-01-01 00:02:21.768	0.595469	0.760637	0.685421	0.578333	0.498498	0.475308
1970-01-01 00:02:21.788	0.605580	0.688843	0.744868	0.555881	0.509187	0.495353
1970-01-01 00:02:21.808	0.645752	0.716176	0.796460	0.609551	0.449418	0.459763

[186 rows x 6 columns])

('NAG',

ts	a-x	a-y	a-z	g-x	g-y	g-z
1970-01-01 00:02:14.190	0.605290	0.604410	0.633610	0.645668	0.413584	0.450158
1970-01-01 00:02:14.210	0.517338	0.560358	0.654610	0.596756	0.416482	0.391182
1970-01-01 00:02:14.230	0.468309	0.572697	0.648463	0.560398	0.400410	0.386810
1970-01-01 00:02:14.249	0.508778	0.674468	0.570546	0.557059	0.463209	0.438607
1970-01-01 00:02:14.269	0.505977	0.635156	0.641031	0.588388	0.499552	0.480875
...	...	...	...	...	...	...
1970-01-01 00:02:17.744	0.566541	0.643788	0.700920	0.514526	0.497387	0.410858
1970-01-01 00:02:17.765	0.543710	0.644960	0.680711	0.642752	0.393350	0.438410
1970-01-01 00:02:17.783	0.505388	0.564308	0.601504	0.744954	0.397718	0.490488
1970-01-01 00:02:17.803	0.563372	0.549609	0.523724	0.653590	0.509687	0.466680
1970-01-01 00:02:17.822	0.516132	0.514570	0.633607	0.475705	0.546812	0.428821

[186 rows x 6 columns])

شکل ۳–۷-دو نمونه از پنجره‌هایی به طول ۱۸۶؛ سمت چپ لیبل ترمزگیری شدید سمت راست، رویدادی که ترمزگیری شدید ندارد[1]

[1] مجموعه دادگان مورد استفاده توسط یک تیم پرتقالی طراحی شده و freada_agressiva به معنی ترمز گیری شدید است

۳-۳-۱ محاسبه ماتریس فاصله

از پایان مرحله قبل دیکشنری‌ای از سه ردیف پنجره‌های موقت داریم که برچسب خورده‌اند که به‌نوعی اطلاعات خام محسوب می‌شوند. حال با کمک الگوریتم FastDTW و الگوهایی که در جدول ۳-۳ داریم میزان شباهت هر یک از این پنجره‌ها را با آن الگوها محاسبه می‌کنیم تا بردار ویژگی استخراج شود.

بردار ویژگی در حقیقت بردار فاصله یا میزان عدم شباهت هر یک از پنجره‌های موقت با رویدادهایی است که از پیش برچسب آن‌ها را می‌دانیم. لازم به ذکر است که برای انجام این کار از کتابخانه FastDTW [۲۰] در python۳ استفاده شد. این کتابخانه توانایی محاسبه الگوریتم FastDTW راهمان طور که در بخش قبل مرور شد، دارا است. برای نمونه شیوه محاسبه فاصله‌ی دو سری زمانی از یکدیگر را درشکل ۳–۸ مشاهده می‌کنیم:

```
import numpy as np
from fastdtw import fastdtw

x = np.array([[1,1], [2,2], [3,3], [4,4], [5,5]])
y = np.array([[1,2], [2,2], [2,3]])
distance, path = fastdtw(x, y)
print(distance)
```

```
10.0
```

شکل ۳–۸-نمونه‌ای از استفاده از تابع fastdtw

حال برای درک بهتر فرایند محاسبه فاصله مثال زیر را بررسی می‌کنیم:

در ردیف پنجره‌های موقت به طول ۱۸۶، که به دنبال رویداد «ترمزگیری خطرناک» هستیم، ۸۰۰۰ پنجره به طول ۱۸۶ داریم، که همگی برچسب دارند؛ بر اساس جدول ۳-۳ می‌دانیم که ۱۲ رخداد ترمزگیری خطرناک با طول‌های متفاوت وجود دارند، به‌طور نمونه پنجره ۶۴۴۰ ام که یک رویداد ترمز خطرناک و رویداد ۱۳۷۴ ام که رویداد خطرناک نیست را با این ۱۲ رخداد که از پیش می‌دانیم حساب می‌کنیم نتیجه محاسبه را می‌توان در شکل ۳–۹ مشاهده نمود. همان‌طور که قابل‌مشاهده است میزان فاصله‌ها در نمونه ۶۴۶۰ بسیار کمتر است.

```
Data_DS[186][6460]                    Data_DS[186][1374]

('freada_agressiva',                  ('NAG',
 [('freada_agressiva',                 [('freada_agressiva',
   [133.09777110858676,                  [216.57961965000032,
    131.2434963936938,                    213.78410126717444,
    138.19830255572367,                   221.69319816111718,
    127.37309238098604,                   213.01005092575562,
    127.48227584802419,                   217.66386619219605,
    115.8136528389646,                    211.91937563298194,
    59.148354294534315,                   156.4247810479509,
    60.22048065437508,                    171.57424316286267,
    49.564167770517074,                   149.27498962977822,
    48.641662555749974,                   155.31953863239352,
    44.6293323020319,                     165.91162402429018,
    60.047173458917981])])                167.374917544600351])])
```

شکل ۳-۹-مقایسه فاصله پنجره‌ای حاوی نمونه ترمزگیری (پنجره ۶۴۶۰) خطرناک از رویدادهای اصلی و پنجره‌ای خالی از نمونه ترمزگیری خطرناک (پنجره ۱۳۷۴) از رویدادهای اصلی

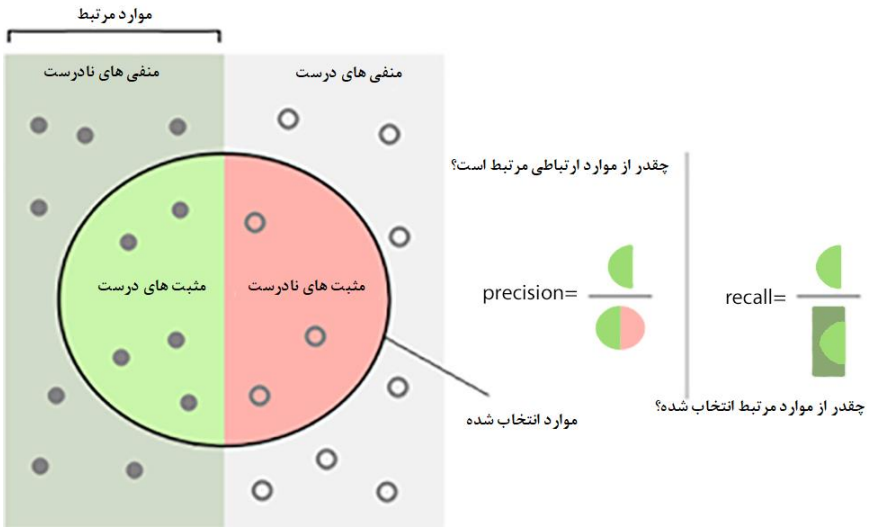
نکته قابل‌توجه این است که محاسبه این مرحله حتی با به‌کارگیری الگوریتم FastDTW بسیار زمان‌بر بوده و زمان اجرای این بخش به تعداد نمونه‌های تولیدشده در مرحله پنجره‌بندی (که با متغیر تصادفی M تعیین می‌شوند) نسبت مستقیم دارد. برای مثال درصورتی‌که متغیر تصادفی M که با میانگین ۰.۵ و احتمال نگاه به رویدادهای پیشین ۲۰ درصد، پنجره‌ها را ایجاد کند، ۳۸۶ دقیقه برای محاسبه این ماتریس فاصله، زمان صرف می‌شود.

۳-۳-۲ پیاده‌سازی مدل یادگیری ماشین

حال وقت آن رسیده که مدلی را انتخاب کرده تا بتواند فرایند برچسب زدن را آموزش ببیند، همچنین در این میان قصد داریم تا مدل ما بتولند، در زمان واقعی و با حداقل تعداد اجرای الگوریتم FastDTW رویداد را تشخیص دهد. از میان روش‌های موجود، تنها الگوریتم‌های درخت تصمیم‌گیری هستند که اجازه می‌دهند مرحله به مرحله بردار ویژگی را حساب کنیم و نیازی به محاسبه‌ی همه‌ی نقاط بردار ویژگی برای یک پنجره‌ی خاص را در یک‌لحظه ندارند. برای مثال درصورتی‌که بخواهیم از مدل‌های SVM یا مدل‌های بیزین یا KNN استفاده کنیم می‌بایست فاصله هر پنجره را با تمامی الگوها حساب کنیم. این در حالی است که در به‌کارگیری مدل‌های درخت تصمیم، این اجازه به ما داده می‌شود که روی درخت به سمت پایین حرکت کرده و با انجام تعداد محدودی الگوریتم FastDTW برای هر سطح از درخت، برچسب پنجره را تشخیص دهیم.

حال در میان مدل‌های درخت تصمیم، از الگوریتم ID۳ استفاده کردیم که هم از تقسیم‌بندی پارامترهای پیوسته و هم از هرس کردن درخت پشتیبانی می‌کند. چراکه بردار ویژگی که از میزان فاصله پنجره‌ها و الگوهای از پیش دانسته به دست آمد، مقادیری پیوسته هستند و باید با کمک روشی که کاندید گیری نامیده می‌شود، کاندیدی که بیشترین بهره اطلاعات را دارد انتخاب و به‌عنوان شرط شاخه برگزیده شود. همچنین با کمک به‌کارگیری هرس کردن درخت که الگوریتم در اختیار ما قرار می‌دهد، این امکان به وجود می‌آید که بتوان شاخه‌هایی از درخت که در صورت حذف کردن آن نتیجه تغییر زیادی نمی‌کند یا نتیجه بهبود می‌یابد را حذف کرده تا نتیجه زودتر حاصل شود و همچنین از overfit شدن جلوگیری کند. برای انجام این کار از کتابخانه decision-tree-id۳ [۲۱]در python۳ استفاده شد. معیارهای مهمی برای ارزیابی اطلاعات وجود دارد که جنبه‌های متفاوتی از مدل را ارزیابی می‌کند در پاراگراف بعد پیش از معرفی مدل تعدادی از این معیارها را مرور می‌کنیم:

۱-دقت[۱] ۲-بازیابی[۲]؛ دقت به معنای درصدی از پیش‌بینی‌های مدل است که مرتبط هستند (چقدر موارد انتخابی درست انتخاب‌شده است؟) و بازیابی به معنای درصدی از کل پیش‌بینی‌هایی است که توسط مدل درست طبقه‌بندی‌شده است (چقدر موارد مرتبط، انتخاب‌شده؟). در شکل ۳—۱۰ این دو معیار به‌سادگی قابل‌مشاهده هستند.



شکل ۳—۱۰-تعریف دقت و بازیابی

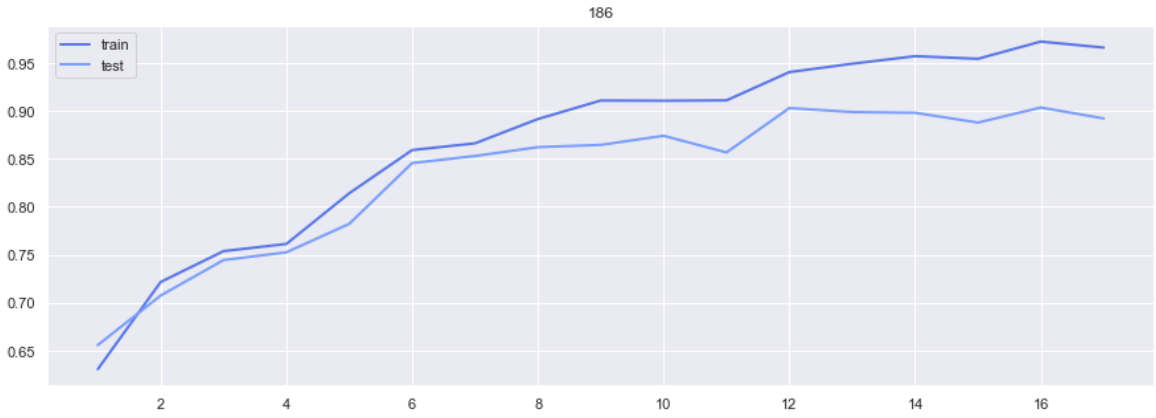
[۱] Precision

[۲] Recall

علاوه به معیارهای فوق که موردبررسی قرار خواهد گرفت، f۱-score نیز به‌عنوان معیاری در رابطه ۳-۸ تعریف می‌شود که مفهومی میانی از هر دو معیار فوق است و می‌تواند معیاری برای دقت کل مجموعه باشد.

$$F_{۱-score} = \frac{۲}{recal^{-۱} + percision^{-۱}}$$ رابطه (۳-۸)

حال به سراغ فرایند آموزش می‌رویم. برای این کار اطلاعات را به دو بخش آموزش و تست، با نسبت ۷۰ به ۳۰ تقسیم‌بندی کرده و فرایند یادگیری را از سر می‌گیریم. ابتدا با دادگان آموزش مدل را آموزش می‌دهیم و با دادگان تست مدل را ارزیابی کرده و برای دقت از معیار F۱-score استفاده می‌کنیم تا میزان ماکزیمم عمق درخت را به ازای هرکدام از پنجره‌ها بهینه کنیم. برای نمونه در شکل ۳-۱۱ می‌توان مشاهده نمود که مدل برای عمق درخت ۱۲ به دقتی با معیار F۱-score به‌صورت ماکرو، عدد ۹۰ درصد برای مجموعه دادگان تست و عدد ۹۴ درصد برای مجموعه دادگان آموزش است. منظور از دقت F۱-score به‌صورت ماکرو این است که این عدد میانگین گرفته از دقت F۱-score برای هریکی از برچسب‌ها است.



شکل ۳-۱۱- نمودار دقت f۱-score به‌صورت ماکرو برای دادگان تست و آموزش

به‌صورت دقیق‌تر اگر بخواهیم بررسی کنیم می‌توانیم، نتایج به ازای عمق درخت ۱۲، برای طبقه‌بندی داده‌های ترمزگیری خطرناک در جدول ۳-۵ را مشاهده کنیم. می‌توان دید که میزان دقت برای یافتن برچسب ترمزگیری خطرناک، میزان ۸۱ درصد بر روی دادگان دادگان تست است.

----------window is 186----------

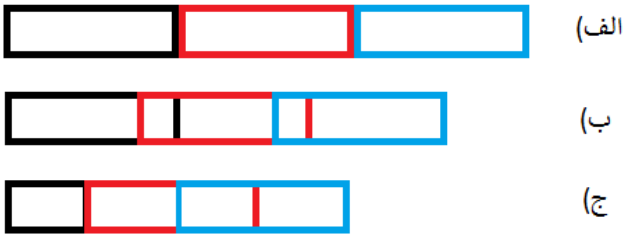
	precision	recall	f1-score	support
NAG	0.99	1.00	1.00	4823
freada_agressiva	0.91	0.73	0.81	114
accuracy			0.99	4937
macro avg	0.95	0.86	0.90	4937
weighted avg	0.99	0.99	0.99	4937

جدول ۳–۵- میزان دقت ترمز برای دادگان تست و آموزش با پنجره ۱۸۶ که با میانگین ۰.۱ و احتمال بازگشت ۱۰ درصد

۳-٤ استفاده از مدل جهت بهره‌برداری

پیش‌تر با مثال، پزشک و یک شخص عادی در تفسیر سری زمانی و نوار قلب آشنا شدیم. حال پس از آموزش مدل، مدل ما مانند پزشکی شده که توانایی تشخیص رویدادها در نوار قلب را دارد. پس اکنون هنگام بررسی سری زمانی، کافی است پنجره، مانند چشم‌پزشک روی سیگنال حرکت کرده و رویدادها را بر اساس آنچه از پیش آموخته موردبررسی قرار داده و برچسب بزند.

هنگام طراحی و استفاده از مدل، باید پنجره‌ای به طول مشخص روی سیگنال حرکت داده تا رویدادها تشخیص داده شوند. دراین‌بین می‌بایست، پارامتر دیگری تحت عنوان λ به مسئله اضافه نمود تا میزان در هم‌تنیدگی و روی‌هم افتادگی پنجره‌های رونده روی سیگنال را کنترل کرد. درصورتی‌که بخواهیم این پارامتر را با مثال پزشک مقایسه کنیم، می‌توانیم این پارامتر را هم‌عرض سرعت حرکت چشم‌پزشک بر روی سری زمانی نوار قلب در نظر بگیریم و سرعت بهینه حرکت چشم‌پزشک روی سیگنال را به دست آوریم، برای این کار پزشک با سرعت‌های متفاوت چشم خود را حرکت داده تا بررسی کند که بهترین سرعت برای تشخیص رویدادها چیست تا ازاین‌پس با آن سرعت، مدل را بررسی کند. حالا با این مثال مشخص می‌شود که برای بهینه‌سازی این پارامتر، می‌بایست سیگنال را به ازای λ های متفاوت پنجره‌بندی کنیم و دقت مدل به‌دست‌آمده از مرحله قبل را برای پنجره‌بندی‌های جدید حساب کنیم. تا λ بهینه را بیابیم. برای درک بهتر این پارامتر در شکل ۳–۱۲ الف) پارامترλ برابر ۱، در ب) λ برابر ۰.۷۵ و در ج) λ برابر ۰.۵ است.



شکل ۳–۱۲- پنجره‌های رونده با پارامتر λ متفاوت هنگام بهره‌برداری مدل

الف) پارامترλ برابر ۱، در ب) λ برابر ۰.۷۵ و در ج) λ برابر ۰.۵ است.

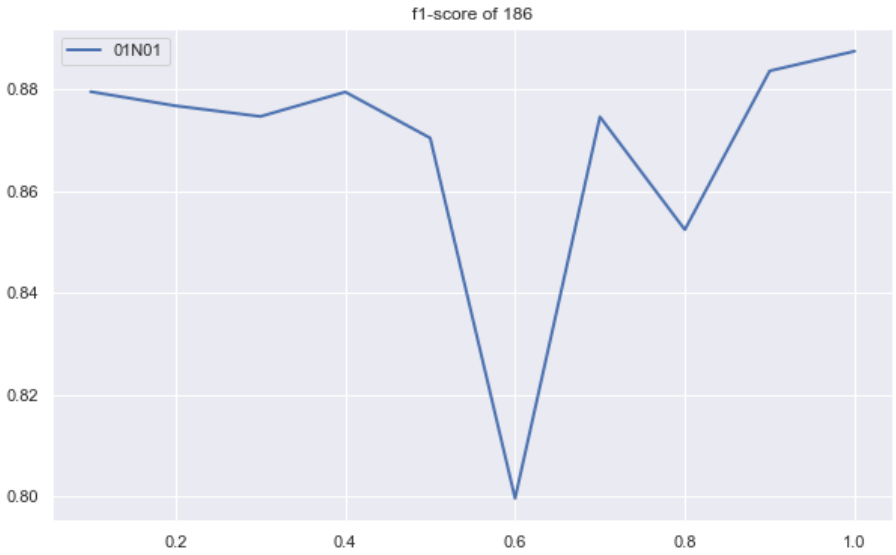
در بخش قبل مدل را با ۷۰ درصد اطلاعات آموزش دادیم و به میزان عمق درخت بهینه رسیدیم. حال که عمق درخت بهینه را یافته‌ایم و تنها می‌خواهیم میزان پارامتر λ را به دست آوریم. مدل را با ۱۰۰ درصد اطلاعات آموزش می‌دهیم، میزان دقت پنجره‌هایی به طول ۱۸۶ بر روی همه‌ی اطلاعات در جدول ۳-۶ قابل‌مشاهده است. دقت این مدل برای رویداد ترمزگیری شدید به عدد ۹۰ درصد رسیده است.

جدول ۳–۶ – میزان دقت ترمز برای کل دادگان با پنجره ۱۸۶ که با میانگین ۰.۱ و احتمال بازگشت ۱۰ درصد

----------window is 186----------

	precision	recall	f1-score	support
NAG	1.00	1.00	1.00	8171
freada_agressiva	0.99	0.83	0.90	209
accuracy			1.00	8380
macro avg	0.99	0.91	0.95	8380
weighted avg	1.00	1.00	1.00	8380

حالا سری زمانی را به شکلی که یاد شد به ازای λ برابر ۰.۱، ۰.۲، ...،۱ پنجره‌بندی کرده و با مدل آموزش داده‌شده‌ی فوق میزان دقت را برای پنجره‌های جدید محاسبه می‌کنیم. نتایج را می‌توان در شکل ۳–۱۳ مشاهده کرد. بر اساس این نمودار می‌توان گفت که به ازای λ=۱ بیشترین دقت برای داده‌های ترمزگیری خطرناک به دست می‌آید، جزئیات دقت این نقطه را در جدول ۳-۷ مشاهده نمود. همان‌طور که در جدول ۳-۷ قابل‌مشاهده است، دقت در این نقطه از عدد ۹۰ درصد به عدد ۷۸ درصد کاهش‌یافته، این عدد دقتی است که در عمل می‌توان از مدل طراحی‌شده توسط ما در فرایند ترمزگیری انتظار داشت. در این میان دو پارامتر دیگر تحت عنوان زمان برای محاسبه و زمان محاسبه قابل‌مشاهده است که در بخش بعدی بیشتر بررسی خواهد شد.



شکل ۳–۱۳- نمودار دقت f۱-score ماکرو برای پنجره به طول ۱۸۶ پس از آموزش کامل دادگان به ازای λ های متفاوت

جدول ۳–۷- بررسی دقیق بهترین نقطه از شکل۳-۱۳

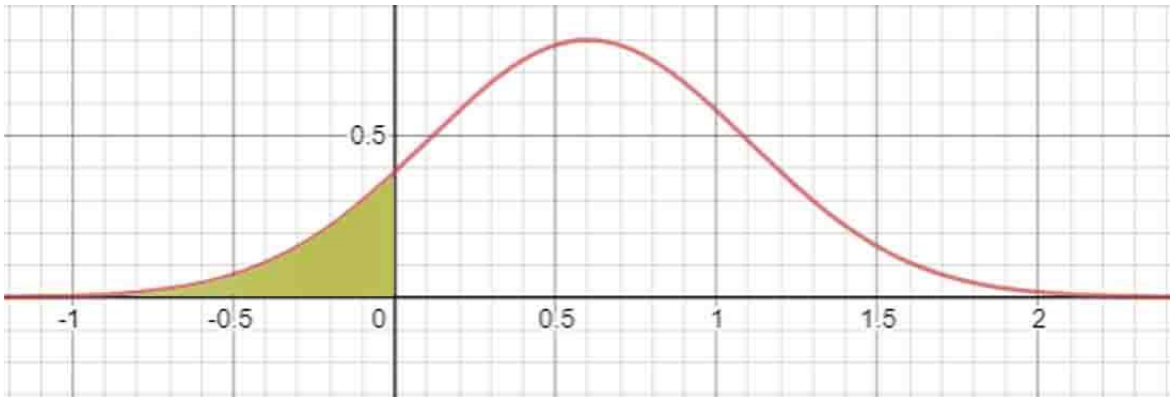
```
==========================================================================
  lambda: 1.00 || window: 186 || line: 01N01 || depth of tree: 14
time to calculatoin : 3.72 || calculation time : 0.6283028270998136
--------------------------------------------------------------------------
                  precision    recall  f1-score   support

              NAG       0.99      1.00      0.99       819
freada_agressiva       0.84      0.73      0.78        22

        accuracy                           0.99       841
       macro avg       0.92      0.86      0.89       841
    weighted avg       0.99      0.99      0.99       841

==========================================================================
```

۳-۵ یافتن بهترین مدل و بررسی زمان اجرا و محدودیت‌ها

در بخش قبل با شیوه کلی آموزش مدل آشنا شدیم دراین‌بین پارامتری به مسئله اضافه شد (متغیر تصادفی M) که توانایی پنجره‌بندی باحالت‌های مختلفی در اختیار ما قرار می‌داد که می‌توانستیم متناسب با آن مدل‌های متفاوتی با دقت و زمان اجراهایی متفاوت در زمان واقعی طراحی کنیم، در این بخش قصد داریم با تغییر در پارامتر پنجره‌بندی مدل‌هایی ایجاد کرده و با بررسی آن‌ها از میان مدل‌هایی که به لحاظ دقت و زمان پردازش توانایی پیاده‌سازی عملی رادارند، بهترین مدل را انتخاب کنیم.

اولین مسئله‌ای که در بخش قبل بررسی شد، متغیر تصادفی M بود که پیشروی پنجره بر روی سری زمانی را همانند چشم انسان مدل می‌کرد، این کار با استفاده از یک توزیع گوسی با میانگین μ و انحراف معیار σ انجام می‌شد که برای درک بهتر از مسئله، بجای استفاده مستقیم از متغیر انحراف معیار، متغیر دیگری تحت عنوان احتمال نگاه به عقب تعریف شد که میزان احتمال عقب رفتن پنجره را انجام می‌داد، تا با کمک آن میزان σ متغیر تصادفی محاسبه شود. اندازه این احتمال با رابطه ۳-۹ تعریف می‌شود و در شکل ۳–۱۴ می‌توان احتمال این پیش‌آمد را به ازای احتمال نگاه به عقب ۲۰ درصد و میانگین ۰.۶، مشاهده نمود، که برای انحراف معیار عدد ۰.۷۵ را به ما برمی‌گرداند.

$$Prob_{neg}(\mu,\sigma) = \int_{-\infty}^{\cdot} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx$$ **رابطه (۳–۹)**



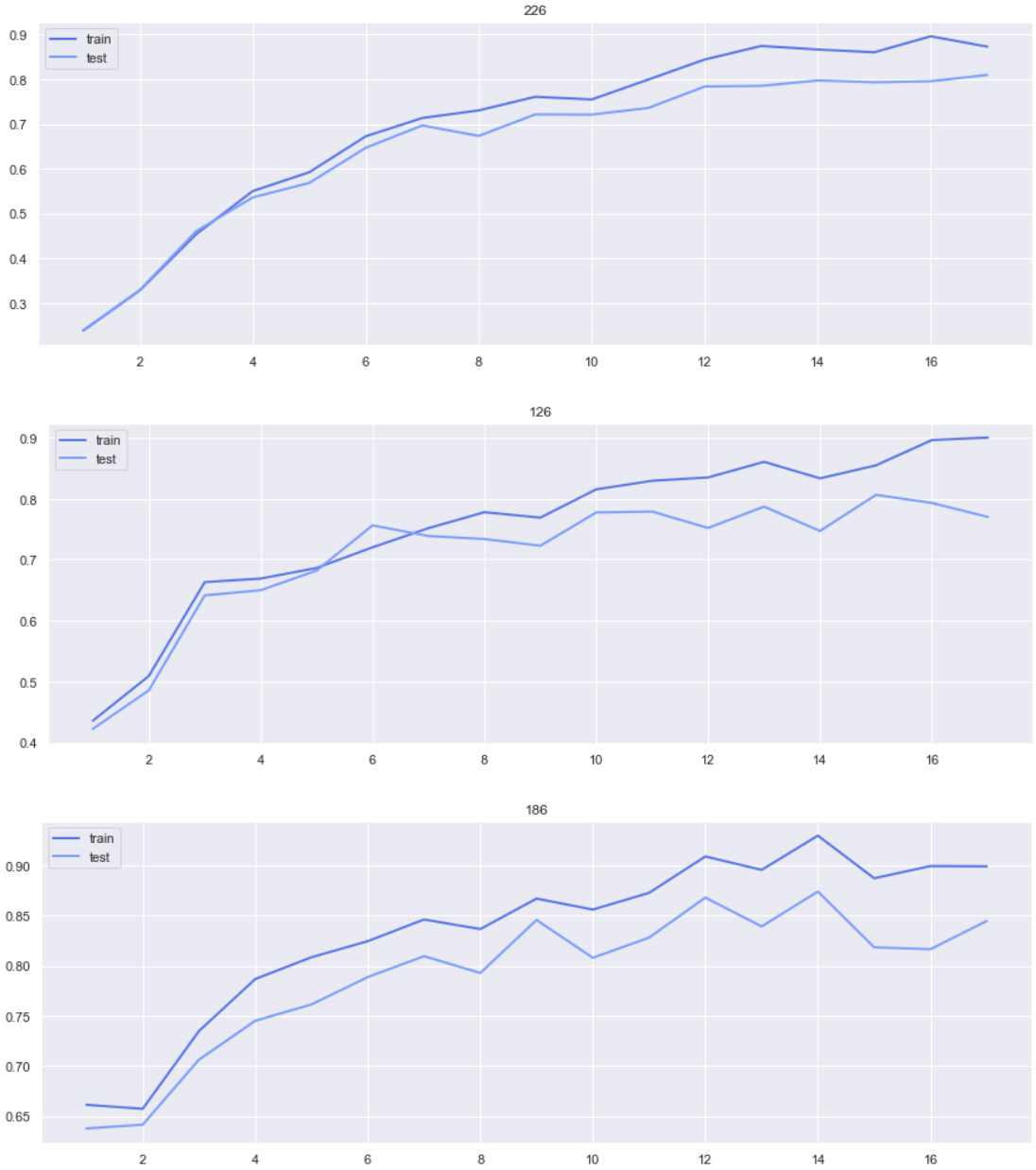
شکل۳–۱۴- پیش‌آمد احتمال نگاه به عقب ۲۰ درصد، با میانگین ۰.۶ که انحراف معیار ۰.۷۵ را برای توزیع برمی‌گرداند.

پیش‌تر گفتیم، میانگین متغیر تصادفی M باید عددی مثبت و کوچک‌تر از یک باشد تا پنجره روبه‌جلو حرکت کند تا بتولند نمونه‌ها را بررسی کند. همچنین احتمال بازگشت به عقب، باید در محدوده‌ای مشخص باشد، اگر از مقداری بیشتر باشد تعداد دیدن نمونه‌ها زیاد می‌شود و درصورتی‌که کم باشد میزان حساسیت آن نسبت به نقاط شروع زیاد می‌شود. به‌طورکلی احتمال بازگشت به عقب، به‌نوعی نماینده انحراف معیار است و انحراف معیار از روی آن حساب می‌شود، با این حساب مسئله را در تعداد نقاط محدودی بررسی می‌کنیم تا درک بهتری از مسئله داشته باشیم. برای این کار جدول ۳-۸ تعریف شد که مسئله را به ازای چندین نقطه بررسی می‌کند. برای این کار ابتدا میزان σ متناسب را به ازای $Prob_{neg}$ و μ های متفاوت به دست می‌آوریم.

جدول ۳–۸ – σ متناسب را به ازای $Prob_{neg}$ و μ های متفاوت

σ		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۰.۰۴	۰.۰۸	۰.۱۲	۰.۱۶
	٪۲۰	۰.۰۶	۰.۱۲	۰.۱۸	۰.۲۴
	٪۳۰	۰.۱	۰.۲	۰.۳	۰.۴

حالا متناسب با مقادیر میانگین و انحراف معیار به‌دست‌آمده از جدول فوق، فرایند پنجره‌بندی روی سیگنال را برای همه‌ی مدل‌هایی که می‌توان ساخت آماده می‌کنیم، سپس برچسب پنجره‌ها را آماده و اطلاعات را به دو قسمت آموزش و تست تقسیم کرده و مدل درختی خود را با عمق درخت‌های متفاوت آموزش می‌دهیم. برای یافتن بهترین عمق درخت، از میزان دقت بر روی دادگان آموزش و تست استفاده می‌کنیم، می‌دانیم میزان دقت برای روی دادگان آموزش با افزایش عمق درخت افزایش می‌یابد تا جایی که اگر عمق درخت از حدی بیشتر شود، مدل تمامی نمونه‌ها را یاد گرفته و دقت مدل روی دادگان آموزش به عدد صد می‌رسد، این در حالی است که میزان دقت روی دادگان تست کاهش می‌یابد، پس باید به دنبال نقطه‌ای باشیم که میزان آموزش روی دادگان تست بهینه باشد. در این مسئله علاوه بر دقت بر روی دادگان تست چون قصد داریم مدل را به‌صورت زمان واقعی، پیاده‌سازی کنیم و زمان برچسب زدن برای پنجره‌ها اهمیت دارد، شرط محدودکننده دیگری را در نظر می‌گیریم، با فرض اینکه زمان تجمیع اطلاعات در پنجره‌ها را ۵ ثانیه و زمان اجرای هر بار الگوریتم DTW، ۱۰۰ میلی‌ثانیه در نظر باشد، با در نظر گرفتن این نکته که به‌طور هم‌زمان باید سه بار فرایند برچسب زدن انجام شود بیشترین میزان عمق درخت عدد ۱۷ می‌خواهد. با مثالی مسئله را بهتر بررسی می‌کنیم؛ مدلی با میانگین ۰.۰۵ و احتمال بازگشت به عقب آن را ۱۰ درصد در نظر بگیرید، در شکل ۳–۱۵ نمودار دقت بر روی دادگان تست و آموزش قابل‌مشاهده است.



شکل ۳–۱۵- نمودار میزان دقت به ازای عمق درخت‌های متفاوت، برای پنجره‌هایی با طول‌های متفاوت.[1]

همان‌طور که در شکل ۳–۱۵ مشاهده می‌کنید، میزان عمق درخت بهینه برای پنجره‌ها به طول ۲۲۶، عدد ۱۴، پنجره با طول ۱۲۶، عدد ۶ و پنجره با طول ۱۸۶، عدد ۹ در نظر گرفته می‌شود. از بخش قبل به یاد داریم که در پنجره‌ها با طول ۲۲۶ به دنبال یافتن رویدادهای شتابگیری خطرناک، گردش‌به‌راست خطرناک، گردش‌به‌چپ خطرناک هستیم. در جدول ۳-۹-الف می‌توان میزان دقت برای هرکدام از این رخدادها را

[1] در بالای هر شکل عدد متناسب به پنجره مربوطه ذکر شده.

مشاهده نمود. به‌طور مشابه نتیجه مدل را می‌توان برای پنجره به طول ۱۲۶، برای رویدادهای تعویض لاین به سمت راست خطرناک و تعویض لاین به سمت چپ خطرناک انجام داد که نتیجه آن را می‌توان درجدول ۳-۹-ب مشاهده کرد. همچنین در پنجره‌هایی به طول ۱۸۶ می‌توان به دنبال رویداد ترمزگیری بود؛ که نتایج دقت آن در جدول ۳-۹-ج قابل‌مشاهده است.

جدول ۳-۹-اطلاعات دقیق میزان دقت نقاط برگزیده‌شده در شکل ۳-۱۵ برای پنجره ها با طول های ۲۲۶، ۱۸۶،۱۲۶

الف

window is 226	precision	recall	f1-score	support
NAG	0.97	0.99	0.98	3762
aceleracao_agressiva	0.76	0.56	0.65	131
curva_direita_agressiva	0.94	0.73	0.82	104
curva_esquerda_agressiva	0.88	0.64	0.74	108
accuracy			0.96	4105
macro avg	0.89	0.73	0.80	4105
weighted avg	0.96	0.96	0.96	4105

ب

window is 126	precision	recall	f1-score	support
NAG	0.99	1.00	1.00	7297
troca_faixa_direita_agressiva	0.84	0.59	0.69	63
troca_faixa_esquerda_agressiva	0.83	0.65	0.73	46
accuracy			0.99	7406
macro avg	0.89	0.75	0.81	7406
weighted avg	0.99	0.99	0.99	7406

ج

window is 186	precision	recall	f1-score	support
NAG	0.99	1.00	0.99	4882
freada_agressiva	0.90	0.65	0.75	137
accuracy			0.99	5019
macro avg	0.94	0.82	0.87	5019
weighted avg	0.99	0.99	0.99	5019

حال با توجه به بررسی فوق که برای یکی از مدل‌ها انجام گرفت، در جدول ۳-۱۰، می‌توان میزان دقت مدل بر روی دادگان تست به ازای شیوه‌های متفاوت پنجره‌بندی مشاهده کرد. در جدول زیر بهترین مدل‌هایی که در این بخش می‌توان بررسی نمود بارنگ قرمز مشخص شد. مقصود از بهترین مدل، مدلی است که بیشترین میزان دقت بر روی دادگان تست را ارائه می‌کند.

جدول ۳-۱۰-میزان دقت به ازای مدل‌های آموزش داده‌شده، با پارامترهای پنجری بندی جدول ۳-۸، به ازای پنجره‌های متفاوت.

F۱-score Macro avg on ۲۲٦		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۸۰	۷۲	۷۰	۷۰
	٪۲۰	۸۳	۷۳	۷۴	۷۴
	٪۳۰	۸۴	۸۱	۸۱	۷۷
F۱-score Macro avg on ۱۲٦		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۸۱	۶۷	۷۷	۶۴
	٪۲۰	۷۶	۷۰	۶۹	۶۹
	٪۳۰	۷۲	۷۴	۷۹	۶۹
F۱-score Macro avg on ۱۸٦		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۸۷	۸۸	۸۲	۸۲
	٪۲۰	۸۸	۸۵	۷۷	۸۵
	٪۳۰	۹۰	۸۸	۸۵	۸۳

نتیجه‌گیری جامعی که می‌توان از جدول فوق به دست آورد این است که هر چه میزان پیشروی ما روی سیگنال کمتر باشد (با μ کوچک‌تری سیگنال را آموزش دهیم) می‌توانیم مدل بهتری آموزش دهیم؛ و برای تشخیص رویدادهای تغییرلاین اگر نگاه به عقب کمتری داشته باشیم مدل بهتر آموزش می‌بینند ولی برای تشخیص سایر رویدادها اگر نگاه به عقب بیشتری داشته باشیم، می‌توانیم مدل‌هایی با دقت بالاتر آموزش دهیم. در جدول ۳-۱۱ می‌توانیم نتایج برای هر یک از نوع رویدادها را برای بهترین مدل مشاهده کنیم.

جدول ۳-۱۱- اطلاعات دقیق میزان دقت نقاط برگزیده‌شده در جدول ۳-۱۰

الف

window is 226	precision	recall	f1-score	support
NAG	0.98	0.99	0.98	3736
aceleracao_agressiva	0.78	0.62	0.69	128
curva_direita_agressiva	0.91	0.82	0.87	90
curva_esquerda_agressiva	0.95	0.74	0.83	98
accuracy			0.97	4052
macro avg	0.90	0.80	0.84	4052
weighted avg	0.97	0.97	0.97	4052

ب

window is 126	precision	recall	f1-score	support
NAG	0.99	1.00	1.00	7297
troca_faixa_direita_agressiva	0.84	0.59	0.69	63
troca_faixa_esquerda_agressiva	0.83	0.65	0.73	46
accuracy			0.99	7406
macro avg	0.89	0.75	0.81	7406
weighted avg	0.99	0.99	0.99	7406

ج

window is 186	precision	recall	f1-score	support
NAG	0.99	1.00	1.00	4823
freada_agressiva	0.91	0.73	0.81	114
accuracy			0.99	4937
macro avg	0.95	0.86	0.90	4937
weighted avg	0.99	0.99	0.99	4937

در جدول ۳-۱۱-الف، می‌توان میزان دقت رویدادهای شتابگیری خطرناک و گردش خطرناک به سمت چپ و راست را بررسی کرد که میزان دقت درست تشخیص دادن رویدادهای گردش به چپ‌خطرناک ۹۵ درصد است که به این معنی که ۹۵ درصد از رویدادهایی که ترمزگیری تشخیص داده‌شده‌اند، واقعا رویداد ترمزگیری خطرناک بوده‌اند. همچنین میزان دقت بازیابی عدد ۷۴ است، به این معنی که ۷۴ درصد از پنجره‌های ترمزگیری تشخیص داده‌شده‌اند، به‌طور مشابه میزان دقت برای گردش‌به‌راست ۹۱ درصد و میزان بازیابی عدد ۸۴ درصد است. میزان دقت برای رویدادهای شتابگیری ۷۸ درصد و بازیابی آن ۶۲ درصد است. به همین منوال میزان دقت داده‌های تغییرلاین را می‌توانیم در جدول ۳-۱۱-ب مشاهده کنیم که میزان دقت در گردش‌به‌چپ خطرناک ۸۳ درصد که میزان بازیابی آن ۶۵ درصد و میزان دقت

در گردش‌به‌راســت خطرناک ۸۴ درصــد که با بازیابی ۵۹ درصــد مدل شــده اســت؛ و در پنجره ســوم، رویدادهای ترمزگیری با دقت ۹۱ درصد و بازیابی ۷۳ درصد، شناسایی شدند.

پس‌ازاین مرحله از آموزش و یافتن بهترین عمق درخت می‌بایســت مدل را با همه‌ی اطلاعات موجود آموزش دهیم تا به یافتن بهترین پارامتر λ برای پیاده‌ســازی در زمان واقعی بپردازیم. در جدول ۳-۱۲ می‌توانیم میزان دقت بهترین مدل‌های به‌دســت‌آمده به ازای پنجره‌های متفاوت، بر روی خود دادگان آموزش مشاهده کنیم، واضح است که میزان دقت باید زیاد باشد، چراکه اطلاعات را روی همان چیزی که آموزش می‌دهیم تست می‌کنیم، هدف از این کار این است که بررسی کنیم به ازای چه میزان λ میزان دقت بیشینه است. تا در عمل هنگام پیاده‌سازی مدل پنجره را با همان سرعت روی نمودار حرکت دهیم.

جدول ۳-۱۲-میزان دقت بهترین مدل برای روی دادگان آموزش

الف

---window is 226---
---

	precision	recall	f1-score	support
NAG	0.98	1.00	0.99	12357
aceleracao_agressiva	0.95	0.71	0.81	462
curva_direita_agressiva	0.98	0.94	0.96	333
curva_esquerda_agressiva	0.96	0.88	0.92	353
accuracy			0.98	13505
macro avg	0.97	0.88	0.92	13505
weighted avg	0.98	0.98	0.98	13505

ب

---window is 126---

	precision	recall	f1-score	support
NAG	0.99	1.00	1.00	7297
troca_faixa_direita_agressiva	0.84	0.59	0.69	63
troca_faixa_esquerda_agressiva	0.83	0.65	0.73	46
accuracy			0.99	7406
macro avg	0.89	0.75	0.81	7406
weighted avg	0.99	0.99	0.99	7406

ج

---window is 186---
---

	precision	recall	f1-score	support
NAG	1.00	1.00	1.00	16057
freada_agressiva	0.98	0.81	0.89	399
accuracy			1.00	16456
macro avg	0.99	0.91	0.94	16456
weighted avg	1.00	1.00	0.99	16456

در بخش قبل به‌طور کامل پارامتر λ تعریف شد، در اینجا تنها به دنبال بررسی بهترین مدل هستیم. در شکل ۳–۱۶-الف برای پنجره‌ای به طول ۲۲۶، به ازای λ برابر ۰.۸، مدلی است که بیشترین دقت را در اختیار ما قرار می‌دهد، همچنین در شکل ۳–۱۶-ب برای پنجره به طول ۱۲۶، به ازای λ برابر ۰.۸ و ۰.۴ دو کاندید برای بهترین میزان دقت در اختیارمان قرار می‌دهد و با بررسی شکل ۳–۱۶-ج برای پنجره به طول ۱۸۶ چهار کلندید به ازای λهای برابر ۰.۱، ۰.۲،۰.۴ و ۰.۵ داریم. در ادامه با تحلیل زمانی میان این کلندیدها بهترین کلندید برای هر پنجره را انتخاب می‌کنیم. از جدول ۳-۱۲ می‌دانیم، میزان دقت ماکرو f۱-score، به‌طور مثال برای پنجره‌هایی با طول ۲۲۶، ۹۲ درصد هنگام پیاده‌سازی در عمل به ازای λ برابر ۰.۸ این عدد به ۹۱ درصد رسید. علت این تفاوت در شیوه نگاه و پنجره‌بندی است، ولی این میزان کاهش اندک نشان از میزان فراگیر بودن مدل است و نشان می‌دهد که نمونه‌ها به‌صورت مناسبی آموزش‌دیده‌اند.

ب

الف

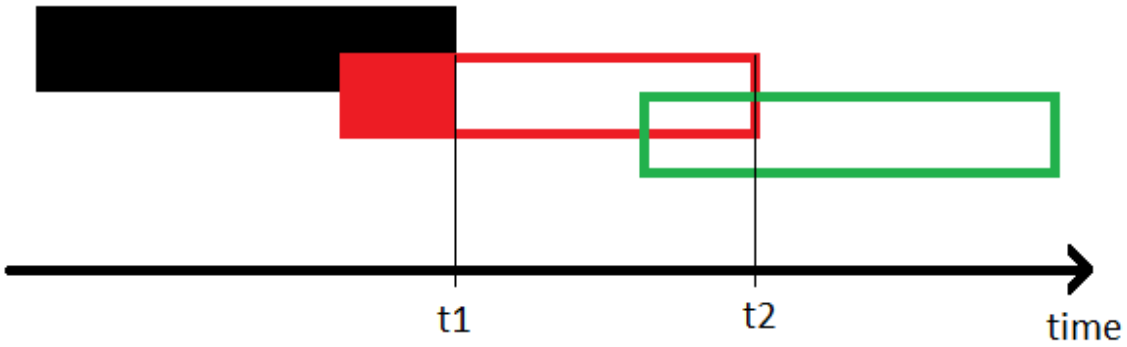
ج

شکل ۳–۱۶-نمودار دقت بهترین مدل‌ها برحسب λ های متفاوت، برای طول های متفاوتی از پنجره

برای بررسی دقیق هر یک از این کاندیدها می‌بایست پردازش در زمان واقعی آن‌ها را بررسی کنیم. می‌دانیم پارامتر λ میزان در هم تنیدگی پنجره‌ها مشخص می‌کند. با دانستن نرخ نمونه‌برداری از اطلاعات، می‌توان زمانی که طول می‌کشد تا پنجره پر شود و سیستم بیکار[1] را محاسبه نمود. شکل ۳–۱۷ را در نظر بگیرید؛ در این شکل λ برابر ۰.۷۵ در نظر گرفته‌شده به این معنا است که تنها ۲۵ درصد از پنجره قبلی در پنجره‌های جدید مورداستفاده قرار می‌گیرند؛ بنابراین زمانی که طول می‌کشد تا پنجره پر شود از رابطه ۳-۱۰ محاسبه می‌شود:

$$\Delta t_{\text{زمان‌انتظار}} = t2 - t1 = \frac{\text{طول پنجره} * \lambda}{\text{فرکانس نمونه برداری}}$$

رابطه (۳–۱۰)



شکل ۳–۱۷-فرایند پنجره‌بندی به ازای λ برابر ۰.۷۵

در تحلیل زمان واقعی علاوه بر مسئله فوق، مسئله مهم محاسبه و برچسب زدن، پنجره قبل است. میدانیم زمان‌برترین بخش انجام محاسبه در فرایند برچسب‌گذاری، اجرای الگوریتم DTW است. تعداد مراحل اجرای این الگوریتم، حداقل برابر تعداد عمق درخت است، چراکه در هر مرحله دقیقا یک‌بار این الگوریتم بین پنجره‌ی موردبررسی و نمونه‌هایی که درخت ذخیره‌شده، اجرا می‌شود. برای این کار باید میزان زمان اجرای این الگوریتم را به ازای طول پنجره‌های متفاوت انجام دهیم و میانگین زمان اجرای آن را روی سخت‌افزار موجود به دست آوریم. این تست روی cpu intel core i5 Gen8 انجام شد که نتایج جدول ۳-۱۳ را به همراه داشت. مشاهده می‌شود که میزان زمان پردازش بر روی هر یک از این سخت‌افزارها زیر ۱۰۰ میلی‌ثانیه است.

[1] idle

جدول ۳-۱۳- زمان پردازش پنجره ها به ازای اجرای الگوریتم DTW

	پنجره به طول ۱۲۶	پنجره به طول ۱۸۶	پنجره به طول ۲۲۶
CPU	۰/۰۳۴۰۲۶۴۷۱۹۰۰۹۷۸	۰/۰۴۴۸۷۸۷۷۳۳۶۴۲۷۲	۰/۰۵۷۵۱۳۹۷۰۷۸۲۶۶۰

حال متناسب با عمق درخت بکار گرفته‌شده و میانگین زمان اجرای هر بار الگوریتم DTW می‌توان فرایند برچسب زدن مدل‌های آموزش‌یافته را در زمان واقعی ارزیابی کرد، برای این کار، زمان محاسبه نتیجه را برای کاندیدهایی که از بخش قبل به دست آمد بر اساس عمق درختی که مدل آن‌ها آموزش‌دیده تخمین می‌زنیم؛ برای مثال اگر عمق درخت مدلی که در پنجره‌ای به طول ۲۲۶ به دنبال رویداد چرخش به چپ خطرناک است ۱۰ باشد، زمان اجرای آن تقریباً ۵۷۰ میلی‌ثانیه در نظر گرفته می‌شود.

حال با توجه به توضیحات فوق و رابطه ۳-۱۰، مثال فوق را می‌توان ازنظر پیاده‌سازی عملی بررسی کرد؛ اگر مدل با طول پنجره ۲۲۶، λ برابر ۰.۱ داشته باشد و فاصله زمانی بین هر نمونه ۰.۰۲ ثانیه باشد، بدین معناست که ما ۴۵۲ میلی‌ثانیه وقت برای پر شدن پنجره و محاسبه برچسب پنجره قبلی داریم، این درحالی‌که است که فرایند برچسب زدن بر روی این پنجره ۵۷۰ میلی‌ثانیه زمان می‌برد، پس‌ازاین مدل نمی‌توان در عمل استفاده نمود. علاوه بر این میزان زمان باید به‌گونه‌ای باشد که تمام اسلات‌های زمانی پر نشود، چراکه به‌صورت موازی دو پردازش دیگر در حال انجام است.

حال با در نظر گرفتن امکان پیاده‌سازی عملی کاندیدهایی که در بخش قبل معرفی کردیم را بررسی می‌کنیم تا بهترین مدل که امکان پیاده‌سازی عملی دارد را بیابیم. برای این کار ابتدا تعدادی از مدل‌ها را محدود کرده تا بتوانیم مدل‌های موجود را آنالیز کرده و در کنار یکدیگر قرار دهیم.

با پنجره به طول ۲۲۶ شروع می‌کنیم، میدانیم برای این پنجره به دلیل دقت بالا تنها یک کلندید با λ برابر ۰.۸ داریم که در جدول ۳-۱۴ می‌توان جزئیات آن را مشاهده کرد. میزان زمانی که طول می‌کشد تا لیبل مشخص شود ۸۶۲ میلی‌ثانیه است و ۳.۶۸۰ ثانیه زمان‌داریم تا اطلاعات در این پنجره تجمیع شوند. قابل‌درک است که هیچ محدودیتی وجود ندارد و به‌سادگی می‌توان از این مدل برای استفاده در زمان واقعی استفاده کرد چراکه تنها ۲۳ درصد از زمان مشغولیت هسته مرکزی را استفاده می‌کند که چون این عدد از ۳۳ درصد (سهم زمانی پردازش برای این رشته از محاسبات) کمتر است، تداخلی برای سایر رشته محاسباتی ایجاد نخواهد کرد.

جدول ۳-۱۴-نتایج بهترین مدل برای پنجره‌هایی به طول ۲۲۶

```
======================================================================
  lambda: 0.80 || window: 226 || line: 005N03 || depth of tree: 15
time to calculation : 3.62 || calculation time : 0.8627095617399083
----------------------------------------------------------------------
                          precision    recall  f1-score   support

                      NAG       0.99      0.99      0.99       798
    aceleracao_agressiva        0.95      0.73      0.83        26
 curva_direita_agressiva        0.95      0.91      0.93        23
curva_esquerda_agressiva        0.91      0.87      0.89        23

                 accuracy                           0.98       870
                macro avg       0.95      0.88      0.91       870
             weighted avg       0.98      0.98      0.98       870

======================================================================
```

به سـراغ پنجره‌هایی به طول ۱۲۶ می‌رویم؛ می‌دانیم در این پنجره به دنبال یافتن تغییرلاین خطرناک هستیم، کاندیدهایی که در این پنجره‌بندی داریم λ هایی برابر ۰.۴ و ۰.۸ دارند، نتایج این مدل به ازای پارامترهای فوق را می‌توان در جدول ۳-۱۵ بررسی کرد، با توجه به این نتایج می‌توان به این نکته دقت کرد که زمان بیشینه برای محاسبه برچسب پنجره‌ها، ۵۱۰ میلی‌ثانیه است و زمان برای محاسبه‌ی برچسب‌ها با توجه به λ های ذکرشده به ترتیب ۱.۰۱۰ و ۲.۰۲۰ است. با توجه به اعداد فوق میدانیم که هر دو مدل قابلیت پیاده‌سازی به‌صورت تک‌رشته‌ای رادارند ولی مدل اول در مجموعه پردازش‌های خود ۵۰ درصد از مواقع هسته مرکزی را در اختیار قرار می‌گیرد، این درحالی‌که است که به ازای λ برابر ۰.۸، تنها ۲۵ درصد از مواقع هسته مرکزی را به پردازش مشغول می‌کند و همچنین این مدل، مدلی است که دقت بهتری دارد، پس آن را به‌عنوان بهترین مدل برای پنجره‌هایی به طول ۱۲۶ انتخاب می‌کنیم.

بعد از بررسی دو مدل فوق، به سراغ پنجره‌هایی به طول ۱۸۶ می‌رویم. در این پنجره تنها به دنبال یافتن ترمزگیری خطرناک هستیم، کلندیدهایی که در این پنجره‌بندی داریم λهای برابر ۰.۱، ۰.۲،۰.۴ و ۰.۵ دارند، با بررسی این مدل در جدول ۳-۱۶ و با توجه به این‌که مدل، عمق درختی برابر ۱۲ دارد. بیشینه زمان برچسب زدن پنجره‌ها ۵۳۸ میلی‌ثانیه است. و زمان محاسبه‌ی برچسب‌ها با توجه به λ های ذکرشده به ترتیب، ۳۷۰، ۷۴۰، ۱۴۹۰ و ۱۸۶۰ میلی‌ثانیه است. با توجه به اعداد فوق میدانیم که مدل اول حتی قابلیت پیاده‌سازی به‌صورت تک‌رشته‌ای را ندارد و در صورت استفاده از مدل دوم، ۷۰ درصد از هسته مرکزی را مشغول می‌کنیم که فرایند برچسب زدن سایر پردازش‌ها را دچار اخلال می‌کند ولی پنجره‌بندی با λهای ۰.۴ و ۰.۵ به ترتیب ۳۶ درصد و ۲۸ درصد از هسته مرکزی را درگیر می‌کنند که

اعدادی قابل قبول اســت. حال برای انتخاب میان این دو مدل، به بررســی دقت آن‌ها می‌پردازیم. مدل با پنجره‌بندی متفاوت برای دادگان ترمزگیری به ترتیب دقت F۱-score، ۸۸ درصـد و ۸۹ درصـد را دارند، پس مدل با λ برابر ۰.۵ را انتخاب می‌کنیم که هم دقت بیشتر و هم‌زمان برای پردازش بیشتری در اختیار ما قرار می‌دهد.

جدول ۳–۱۵- نتایج بهترین کاندیدها برای پنجره‌هایی به طول ۱۲۶

الف

```
==================================================================
  lambda: 0.40 || window: 126 || line: 005N01 || depth of tree: 15
time to calculation : 1.01 || calculation time : 0.5103970785146787
------------------------------------------------------------------
                              precision    recall  f1-score   support

                         NAG       1.00      1.00      1.00      3086
 troca_faixa_direita_agressiva       0.83      0.62      0.71        24
troca_faixa_esquerda_agressiva       0.68      0.72      0.70        18

                    accuracy                           0.99      3128
                   macro avg       0.84      0.78      0.80      3128
                weighted avg       0.99      0.99      0.99      3128

==================================================================
```

ب

```
==================================================================
  lambda: 0.80 || window: 126 || line: 005N01 || depth of tree: 15
time to calculation : 2.02 || calculation time : 0.5103970785146787
------------------------------------------------------------------
                              precision    recall  f1-score   support

                         NAG       1.00      1.00      1.00      1545
 troca_faixa_direita_agressiva       0.90      0.75      0.82        12
troca_faixa_esquerda_agressiva       0.58      0.88      0.70         8

                    accuracy                           0.99      1565
                   macro avg       0.83      0.87      0.84      1565
                weighted avg       0.99      0.99      0.99      1565

==================================================================
```

۳-۶ جمع‌بندی

برای جمع‌بندی باید به این نکته اشـاره کرد که با کمک مدل آموزشـی که در بخش قبل بررسـی کردیم، یک مدل برای هر پنجره انتخاب شـــد که در هر پنجره به دنبال یافتن رویدادهایی خاص هســـتیم که جزئیات نتایج آن به شرح زیر است:

در پنجره به طول ۲۲۶، میزان دقت F۱-score برای رویداد شـتاب گیری خطرناک، به عدد ۸۳ درصـد، برای رویداد گردش‌به‌راســت خطرناک ۹۳ درصــد و برای رویداد گردش‌به‌چپ خطرناک عدد ۸۹ درصــد می‌رســیم که محاســبه این کار هر ۳.۸۶ ثانیه و به مدت ۸۶۲ میلی‌ثانیه زمان می‌برد. در پنجره به طول ۱۲۶، میزان دقت F۱-score برای رویداد تعویض لاین به سمت راست خطرناک، به‌دقت ۸۲ درصد، برای

رویداد گردش‌به‌چپ خطرناک ۷۰ درصد می‌رسیم که محاسبه این کار هر ۲.۰۲ ثانیه و به مدت ۵۱۰ میلی‌ثانیه زمان می‌برد. در پنجره به طول ۱۸۶، میزان دقت F۱-score برای رویداد ترمزگیری خطرناک، به‌دقت ۷۸ درصد می‌رسیم که محاسبه این کار هر ۱.۸۶ ثانیه و به مدت ۵۳۸ میلی‌ثانیه زمان می‌برد. حال می‌توانیم مدل‌های طراحی شده فوق را برای به‌کارگیری در عمل مورداستفاده قرار دهیم.

جدول ۳-۱۶- نتایج بهترین کاندیدها برای پنجره‌هایی به طول ۱۸۶

الف

```
================================================================
  lambda: 0.10 || window: 186 || line: 005N03 || depth of tree: 12
time to calculation : 0.37 || calculation time : 0.5385452803712687
----------------------------------------------------------------
                  precision    recall  f1-score   support

             NAG       0.99      1.00      1.00      8475
freada_agressiva       0.89      0.69      0.78       202

        accuracy                           0.99      8677
       macro avg       0.94      0.85      0.89      8677
    weighted avg       0.99      0.99      0.99      8677

================================================================
```

ب

```
================================================================
  lambda: 0.20 || window: 186 || line: 005N03 || depth of tree: 12
time to calculation : 0.74 || calculation time : 0.5385452803712687
----------------------------------------------------------------
                  precision    recall  f1-score   support

             NAG       0.99      1.00      1.00      4124
freada_agressiva       0.88      0.70      0.78        98

        accuracy                           0.99      4222
       macro avg       0.94      0.85      0.89      4222
    weighted avg       0.99      0.99      0.99      4222

================================================================
```

ج

```
================================================================
  lambda: 0.40 || window: 186 || line: 005N03 || depth of tree: 12
time to calculation : 1.49 || calculation time : 0.5385452803712687
----------------------------------------------------------------
                  precision    recall  f1-score   support

             NAG       0.99      1.00      0.99      2063
freada_agressiva       0.84      0.72      0.77        50

        accuracy                           0.99      2113
       macro avg       0.92      0.86      0.88      2113
    weighted avg       0.99      0.99      0.99      2113

================================================================
```

د

```
================================================================
  lambda: 0.50 || window: 186 || line: 005N03 || depth of tree: 12
time to calculation : 1.86 || calculation time : 0.5385452803712687
----------------------------------------------------------------
                  precision    recall  f1-score   support

             NAG       0.99      1.00      1.00      1641
freada_agressiva       0.88      0.70      0.78        40

        accuracy                           0.99      1681
       macro avg       0.93      0.85      0.89      1681
    weighted avg       0.99      0.99      0.99      1681

================================================================
```

٤ فصل چهارم

پیاده‌سازی سیستم دریافت و ذخیره‌سازی اطلاعات

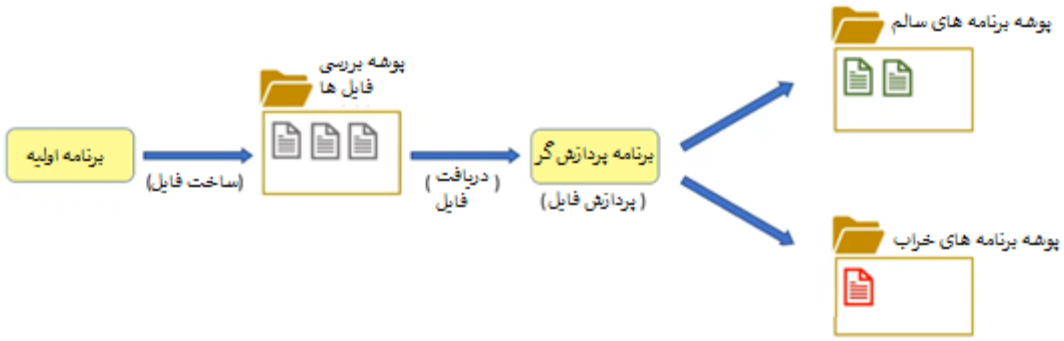
پیاده‌سازی سیستم دریافت و ذخیره‌سازی اطلاعات

در این بخش به پیاده‌سازی بستری برای دریافت اطلاعات می‌پردازیم و سعی می‌کنیم با ابزارهای یکپارچه‌سازی بتوانیم، شرایطی برای دریافت و ذخیره‌سازی اطلاعات فراهم کنیم. در ابتدا مفهوم یکپارچه‌سازی و سپس تعدادی از نرم‌افزارهای مورداستفاده را بررسی کرده و در ادامه نحوه استفاده و ارتباط آن‌ها با یکدیگر شرح داده می‌شود.

۴-۱ مفهوم یکپارچه‌سازی

یک سیستم کامل، از چند برنامه تشکیل‌شده که وظایف متفاوتی را انجام می‌دهند، در اغلب مواقع این برنامه‌ها نیاز دارند با یکدیگر تعامل داشته باشند و باهم ارتباط برقرار کنند. به ایجاد کردن ارتباط میان برنامه‌های متفاوت یکپارچه‌سازی[۱] می‌گویند. شیوه‌های متفاوتی از یکپارچه‌سازی وجود دارد.

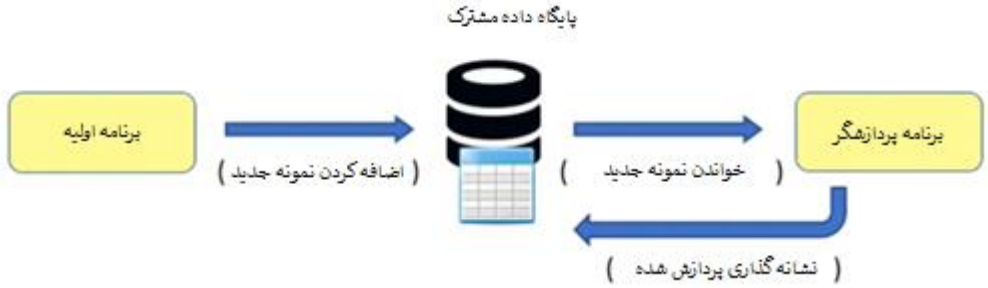
مدل اول یکپارچه‌سازی مبتنی بر فایل است. در این مدل، برنامه اولیه فایلی که باید پردازش شود را ایجاد می‌کند و در پوشه‌ای مشخص قرار می‌دهد، سپس برنامه‌ای دیگر فایل‌هایی که در این پوشه قرارگرفته‌اند را بررسی می‌کند. برای مثال برنامه دوم و فایل‌های خراب را از سالم جدا کرده و در پوشه‌هایی متفاوت قرار می‌دهد. قابل‌مشاهده است که این دو برنامه می‌توانند مستقل از هم کار کنند و حتی می‌توانند با زبان‌های برنامه‌نویسی متفاوتی ایجادشده باشند. در شکل ۴–۱ می‌توان مثال فوق را مشاهده کرد.



شکل ۴–۱-مثالی از یکپارچه‌سازی مبتنی بر فایل

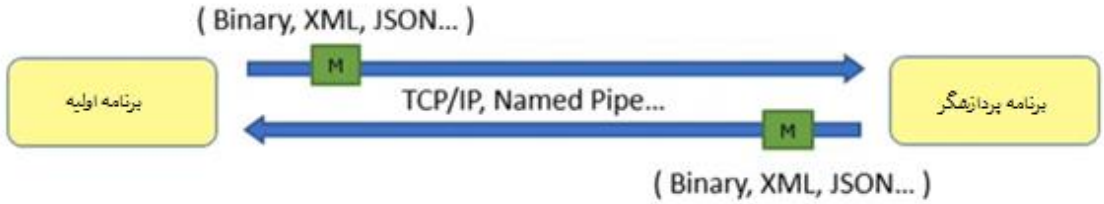
[۱] integration

مدل دوم یکپارچه‌سازی‌هایی مبتنی بر پایگاه‌داده مشترک یا توزیع‌شده است. در این مدل ابتدا یکی از برنامه‌ها اطلاعاتی را در پایگاه‌داده تغییر می‌دهد. سپس برنامه دیگری آن را برداشته و پردازش می‌کند و مجدداً به پایگاه‌داده برمی‌گردلند و به آن برچسب پردازش‌شده می‌زنند. به‌طور مشابه، این دو برنامه می‌توانند مستقل از یکدیگر فعالیت کنند. در شکل ۴–۲ می‌توان مدل فوق را بررسی نمود.



شکل ۴–۲-مثالی از یکپارچه‌سازی مبتنی بر پایگاه‌داده

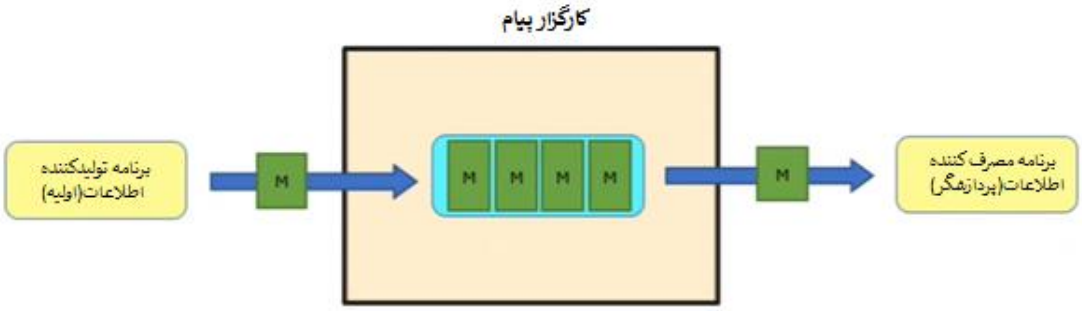
مدل سوم یکپارچه‌سازی‌هایی مبتنی بر ارتباط مستقیم است، به‌نحوی‌که برنامه‌ها به‌صورت مستقیم با یکدیگر ارتباط tcp/ip برقرار کرده و بعد از برقراری ارتباط شروع به ارسال پیام به یکدیگر می‌کنند. فرمت پیام‌ها می‌تواند به گونه‌های متفاوتی ازجمله باینری یا مبتنی برنوشته مانندxml, json , ... باشد. در شکل ۴–۳ می‌توان نمونه‌ای از این ارتباط را مشاهده کرد.



شکل ۴–۳-نمونه از یکپارچه‌سازی بر اساس ارتباط مستقیم میان برنامه‌ها

مدل چهارم یکپارچه‌سازی بر پایه پیام دادن نامتقارن با کمک یک کارگزار پیام[1] است. برنامه اولیه یا تولیدکننده اطلاعات در یک‌سو با هر فرمتی که بخواهد می‌تواند، پیام ارسال کند ولی این باریک برنامه میانی پیام‌ها را دریافت می‌کند (که اغلب به آن message broker یاmessage bus می‌گویند) پیام‌ها را در یک لیست قرار داده که به آن Queue یا صف می‌گویند و آن‌ها را به هدف، مقصد یا مصرف‌کننده منتقل می‌کند. در شکل ۴–۴ می‌توان نمونه‌ای از این شیوه انتقال اطلاعات را مشاهده کرد. این روش، یک روش محبوب است که در سیستم‌های متفاوتی می‌تواند مورداستفاده قرار بگیرد.

[1] Message Broker



شکل ۴-۴-نمونه‌ای از یکپارچه‌سازی با کمک یک کارگزار پیام

این شیوه از یکپارچه‌سازی مزایای بسیاری دارد که به شرح زیر است:

- جدا بودن برنامه تولیدکننده از برنامه مصرف‌کننده:
  - برنامه تولیدکننده اطلاعات و برنامه مصرف‌کننده نیازی ندارد یکدیگر را بشناسند.
  - آدرس و تکنولوژی‌های مورداستفاده یکدیگر را نمی‌دانند.
  - تنها چیزی که نیاز است نسبت به هم آگاه باشند، فرمت اطلاعات ارسالی و دریافتی است.
- ایجاد محیطی مطمئن، قابل‌اعتماد برای انجام پردازش‌ها و ارتباط میان برنامه‌ها:
  - برنامه‌های تولیدکننده اطلاعات می‌تولند به ارسال اطلاعات بپردازد، درحالی‌که هیچ برنامه‌ای در طرف دیگر، اطلاعات را مورداستفاده قرار نمی‌دهد، به این معنا که اطلاعات در برنامه میانی به‌صورت موقت ذخیره می‌شوند و در ادامه هرگاه که برنامه مصرف‌کننده اطلاعات، فعالیت خود را آغاز کرد، می‌تواند از اطلاعات استفاده نماید.
  - درصورتی‌که برنامه مصرف‌کننده در پردازش شکست بخورد، می‌تواند درخواست ارسال مجدد داشته باشد.
- فراهم کردن راهی برای پیاده‌سازی معماری‌هایی افقی در تعداد بالا
  - اگر برنامه مصرف‌کننده اطلاعات به‌تنهایی نتواند همه پیام‌ها را پردازش کند، می‌توان چندین مصرف‌کننده اطلاعات را در کنار هم قرارداد تا فرایند سریع‌تر انجام شود.

این شیوه بسیار پربازده‌تر از مدل‌های یکپارچه‌سازی مبتنی بر پایگاه‌داده، است.

۴-۲ مروری بر داکر

به‌طورکلی داکر[۱] یک سکو برای توسعه، انتقال و اجرای برنامه‌ها است. با کمک داکر می‌توانیم نرم‌افزارها را مستقل از زیرساخت به‌سرعت پیاده‌سازی کنیم. همچنین با کمک این نرم‌افزار می‌توان زیرساخت و سخت‌افزار مورداستفاده در پروژه‌ها را مدیریت کرد و فاصله زمانی میان پیاده‌سازی و رسیدن به مرحله محصول نهایی را به حداقل رساند.

داکر تولنایی بسته‌بندی و اجرای برنامه‌های ما را در محیط ایزوله‌ای به نام کانتینر[۲] فراهم می‌کند. این ایزوله شدن شرایطی ایجاد می‌کند که کانتینرها را بتوان به‌صورت هم‌زمان روی کامپیوتر میزبان اجرا کرد. این کانتینرها در مقابل ماشین‌های مجازی بسیار سبک هستند و می‌توانند بدون نظارت بر روی کامپیوتر میزبان اجرا شوند. به‌طورکلی داکر ابزار و سکویی برای مدیریت کردن این کانتینرها است تا توسعه سریع و ساده‌تر شود و بتوان پردازش قابل توسعه‌ای، روی خوشه‌ای از کامپیوترها اجرا کرد. همچنین هنگامی‌که برنامه کامل شد و به مرحله محصول نهایی رسید، بدون در نظر گرفتن نوع زیرساخت، قابل‌انتقال به محیط‌های دیگر (مرکز داده‌ها، شرکت‌های خدمات دهنده ابری،...) باشد.

به‌طورکلی هسته اصلی داکر از سه بخش تشکیل‌شده که در شکل ۴–۵ قابل‌مشاهده است، بخش اول یک سرویس‌دهنده است که مدیریت اشیا ساخته‌شده توسط داکر را بر عهده دارد، اشیائی همچون فایل ایمیج (فشرده‌شده کانتینرها)، کانتینرها، شبکه‌ها و فضاهایی که تخصیص داده می‌شوند. این برنامه فرایند محاسباتی[۳] نام‌گذاری می‌شود. بخش دوم یک برنامه رابط است (REST API) که دستوراتی که ما می‌دهیم را به فرایند محاسباتی منتقل می‌کند. بخش سوم یک رابط و خط دستور (CLI)[۴] است که فرمان‌های ما در آن وارد می‌شود. خط دستور از یک REST API استفاده می‌کند تا وظیفه کنترل و ارتباط با فرایند محاسباتی را یا با استفاده از فرمان‌هایی از پیش نوشته‌شده یا به‌صورت واردکردن در خط دستور انجام دهد.

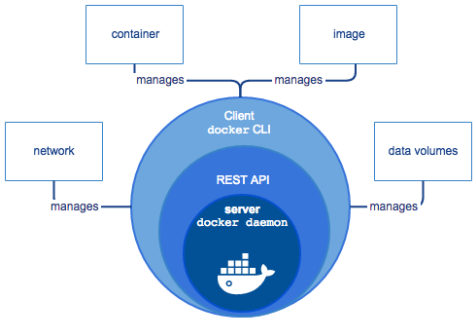
---

۱ Docker

۲ container

۳ Daemon process

۴ Command line



شکل ۴–۵–موتور و هسته داکر که روند فرایند مدیریت برنامه را مشخص می‌کند. [۲۲]

به سه دلیل اصلی در این پروژه برای پیاده‌سازی فرایند دریافت و ذخیره‌سازی اطلاعات از داکر استفاده شد که به شرح زیر است:

پیشرفت و توسعه مداوم برنامه‌ها؛

داکر با کار در محیط‌های استاندارد و با استفاده از کانتینرهای محلی این اجازه را به توسعه‌دهندگان برای پیاده‌سازی برنامه‌های پیچیده می‌دهد و چرخه عمر توسعه را ساده می‌کند. چراکه بخش‌های مختلف برنامه می‌توانند مستقل از هم با نسخه‌ها و حتی زبان‌های برنامه‌نویسی متفاوت پیاده‌سازی و اجرا شوند. به این عمل فرایند یکپارچه‌سازی پیوسته[۱]/ تحویل پیوسته[۲] می‌گویند.

بستر مبتنی بر کانتینرهای داکر، ایجاد برنامه‌هایی قابل‌حمل[۳] را امکان‌پذیر می‌کند. کانتینرهای داکر می‌توانند بر روی کامپیوتر محلی توسعه‌دهنده، در ماشین‌های فیزیکی یا مجازی در یک مرکز داده، بر روی ارائه‌دهندگان خدمات ابری یا در مخلوطی از این محیط‌ها اجرا شوند که کمک می‌کند برنامه به‌سادگی روی کامپیوتر توسعه‌دهنده آماده و سپس برای پیاده‌سازی عملی به سرورهایی باقدرت و پاسخ‌گویی بیشتر منتقل شود.

کانتینرهای داکر سبک و سریع هستند چراکه تنها برنامه‌هایی که موردنیاز است را در خود جای‌داده‌اند. این‌یک جایگزین مناسب و مقرون‌به‌صرفه به‌جای ماشین‌های مجازی است که نیازمند پیش‌نیازهای

---

۱ CI(continuous integration)

۲ CD(continuous delivery)

۳ Portable

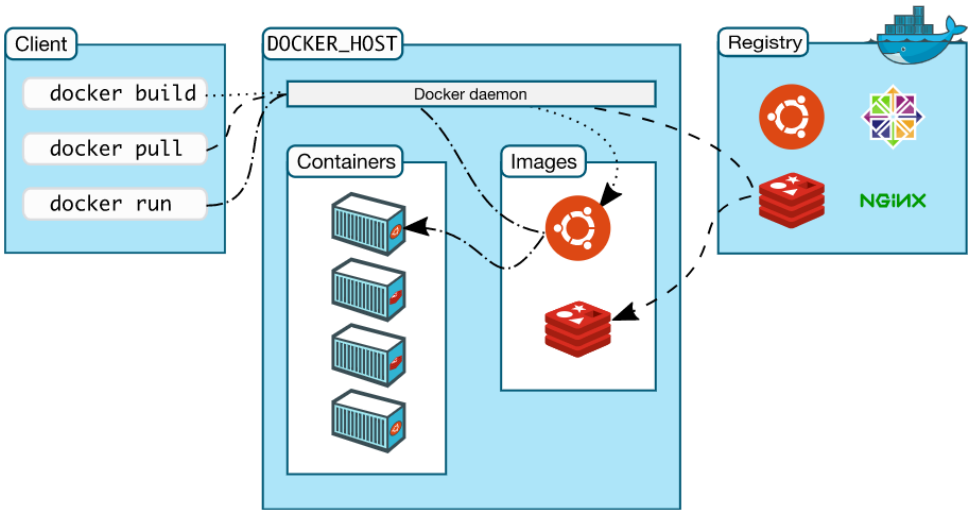
بیشتری برای پیاده‌سازی محیط‌های ایزوله هستند؛ بنابراین می‌توان از ظرفیت محاسبه بیشتری برای رسیدن به اهداف خود استفاده کرد. داکر برای برنامه‌هایی با چگالی بالا (بخش‌های متفاوت) و عملکردهای کوچک و متوسط که نیاز به انجام کارهای بیشتر با منابع کمتری دارند، بسیار مناسب است.

داکر از یک معماری کاربر-سرویس‌دهنده‌ای استفاده می‌کند. به این شکل که کاربر داکر، با فرایند محاسباتی در ارتباط است که وظیفه ساخت و اجرا و توزیع کانتینرها را بر عهده دارد. کاربر داکر و فرایند اصلی می‌توانند در یک کامپیوتر باشند یا می‌توانند از طریق REST API یا رابط‌های شبکه با یکدیگر در ارتباط باشند. به‌طورکلی برنامه‌ها در محیطی به نام کانتینر اجرا می‌شوند که نمونه‌ای قابل‌اجرا از فایلی به نام ایمیج است. برای ساخت فایل ایمیج یا باید خودمان آن فایل را از ابتدا ایجاد کنیم، یا از ایمیج های استانداری که در بخش رجیستری داکر (داکرهاب) قرارگرفته استفاده کنیم. عموماً برای سبک بودن محیط اجرای کار حتی درصورتی‌که بخواهیم فایل ایمیجی را خودمان طراحی کنیم از یک ایمیج مبتنی بر لینوکس ساده شروع کرده و هر چیزی که به آن نیاز داریم را به آن اضافه می‌کنیم، ولی در اکثر مواقع برنامه‌های موردنیاز در داکرهاب وجود دارند که می‌توان از آن بهره‌برد.

برای مثال در شکل ۴–۶ با دستور اول یک فایل ایمیج ساخته می‌شود. با دستور دوم یک فایل ایمیج از بخش رجیستری (داکرهاب) به کامپیوتر میزبان منتقل می‌شود و با دستور سوم، یکی از فایل‌های ایمیج ساخته‌شده به‌صورت کانتینر درآمده و اجرا می‌شود.

به این شکل کانتینرها ایجاد می‌شوند. حال برای ارتباط کانتینرها با یکدیگر از مکانیزم های دیگری که پیش‌تر در شکل ۴–۵ دیدیم استفاده می‌شود. مکانیزم هایی همچون ایجاد شبکه یا اجازه دسترسی به حجم‌های فیزیکی که ارتباط میان کانتینرها را فراهم کند.

همان‌طور که در شکل ۴–۶ قابل‌مشاهده است، برای اجرای هر مرحله می‌بایست دستوری را اجرا کنیم تا مراحل پیش‌رود و برنامه اجرا شود، این کار را می‌توان با کمک داکر کامپوز به‌صورت خودکار انجام داد و روابط بین کانتینرها را مشخص کرد. داکر کامپوز یک ابزار برای تعریف و اجرای برنامه‌های حاوی چند کانتینر است. این برنامه از یک فایل YAML برای پیکربندی این مجموعه استفاده می‌کند. سپس با اجرای تنها یک دستور همه‌ی سرویس‌های خود را فعال و آماده‌به‌کار می‌کند. ساخت برنامه‌ها با داکر کامپوز از سه بخش تشکیل‌شده است.



شکل ۴–۶- نمایی از معماری داکر و فرایند چرخش دستورات و اطلاعات. [۲۲]

بخش اول ساخت Dockerfile است؛ که در حقیقت ایمیجی جدید بر اساس آنچه که در آن واردشده، می‌سازد. برای مثال از ایمیج پایتون که به‌صورت استاندارد در رجیستری (داکرهاب) وجود دارد، شروع کرده و در آن فایل دستور مربوط به نصب کتابخانه‌های موردنیاز را وارد کرده، سپس برنامه پایتون نوشته‌شده را به کانتینر منتقل تا برنامه را در آن اجرا شود. در ادامه در شکل ۴–۷ نمونه‌ای از آن را مشاهده می‌کنیم.

بخش دوم، سرویس‌هایی که است که در فایل docker-compose.yml تعریف می‌شوند که شامل نحوه ارتباط کانتینرها با یکدیگر از طریق شبکه داخلی و ارتباط شبکه کانتینرها با شبکه خارجی (کامپیوتر میزبان) و تنظیم دسترسی‌های حجمی بر روی کامپیوتر میزبان است. در شکل ۴–۸ نمونه‌ای از این فایل را مشاهده می‌کنیم.

```
FROM python:3
WORKDIR /usr/src/app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
COPY main.py ./
COPY util.py ./

CMD [ "python", "./main.py" ]
```

شکل ۴–۷-نمونه‌ای از یک فایل dockerfile

```yaml
networks:
  app-tier:
    driver: bridge
services:
  queue:
    build: rbmq/.
    container_name: rabbitmq
    ports:
      - 5672:5672
      - 15672:15672
      - 1884:1884
    volumes:
      - "C:/Users/hosse/Docker/Rabbitmq/rbmq/Conf/:/etc/rabbitmq/"
    networks:
      - app-tier
  consumer:
    build: sl/.
    container_name: stl
    volumes:
      - "C:/Users/hosse/Docker/Rabbitmq/sl/log/:/usr/src/app/log"
    networks:
      - app-tier
  cache:
    image: redis:latest
    container_name: redis
    ports:
        - 6379:6379
    volumes:
        - C:/Users/hosse/Docker/Rabbitmq/redis/config/redis.conf:/redis.conf
        - C:/Users/hosse/Docker/Rabbitmq/redis/Data:/data
    command: [ "redis-server", "/redis.conf" ]
    networks:
      - app-tier
```

شکل ۴–۸- نمونه‌ای از یک فایل docker-compose.yml

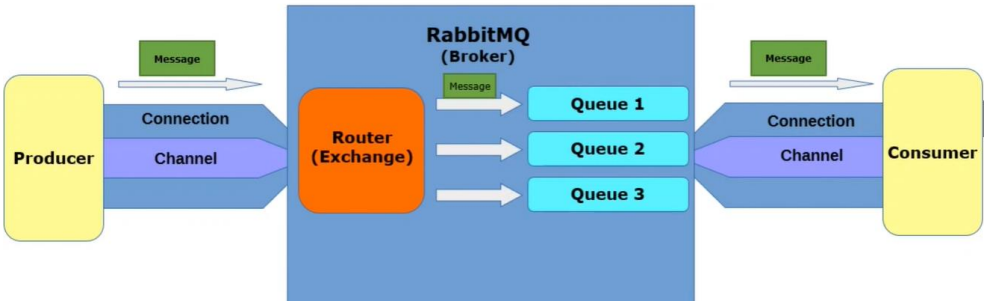
در پایان هم تنها با اجرای دستور docker-compose -up، می‌توان همه ایمیج‌ها را از روی فایل docker-compose.yml، ساخته یا از رجیستری (داکرهاب) دریافت کرد و در ادامه از روی ایمیج‌های ساخته‌شده، کانتینرها ساخته‌شده و برنامه‌ها درون آن‌ها اجرا شوند.

۴-۳ مروری بر کارگزار RabbitMQ

RabbitMq یک کارگزار پیام است که می‌توان با کمک آن سیستم‌های بزرگی را یکپارچه‌سازی کرد. به این شکل که پیام را ابتدا از تولیدکنندگان اطلاعات دریافت می‌کند و به مصرف‌کنندگان اطلاعات منتقل می‌کند. این نرم‌افزار متن‌باز و با زبان برنامه‌نویسی Erlang نوشته‌شده است و از پروتکل‌های بسیاری ازجمله AMQP، STOMP، MQTT، HTTP و WebSocket پشتیبانی می‌کند. دراین‌بین

پروتکل MQTT، یکی از پرکاربردترین پروتکل‌ها در حوزه اینترنت اشیا است، چراکه بسیار سبک است و توانایی پیاده‌سازی بر روی سیستم‌های قابل توسعه را دارد. به همین جهت از این نرم‌افزار می‌توان به‌عنوان پلی میان سخت‌افزار و محیط ابری یادکرد. این نرم‌افزار بر روی ویندوز، لینوکس و مک قابل‌استفاده است و همچنین به‌صورت کانتینر داکر در رجیستری داکر (داکرهاب) قرارگرفته است که استفاده از آن را بسیار ساده می‌کند.

در شکل ۴–۹ معماری کلی این نرم‌افزار قابل‌مشاهده است، تولیدکنندگان اطلاعات[۱] و مصرف‌کنندگان اطلاعات[۲] در دو سمت این معماری و این نرم‌افزار در مرکز شکل قرارگرفته‌اند. آن‌ها به این نرم‌افزار متصل شده و به تبادل اطلاعات می‌پردازند. در این معماری اجزای دیگری ازجمله صف[۳]، مرکز توزیع[۴] و کانال‌ها[۵] قابل‌مشاهده هستند که در ادامه با بررسی یک مثال به شرح وظایف هرکدام از این اجزا می‌پردازیم.



شکل ۴–۹-معماری کلی نرم‌افزار RabbitMQ [۲۳]

یک قطعه اینترنت اشیا را در نظر بگیرید که می‌خواهد اطلاعات خود را با پروتکل MQTT به یک سرویس‌دهنده ارسال کند. می‌دانیم که این پروتکل از چندین کیفیت سرویس‌دهی پشتیبانی می‌کند ولی با فرض استفاده از کیفیت سرویس‌دهی ۱، یک بسته حاوی اطلاعات سنسور را به کارگزار پیام ارسال می‌کند و صبر می‌کند تا کارگزار دریافت این پیام را تصدیق کند. این ارتباط یک ارتباط TCP است و کانال که در شکل ۴–۹ تعریف شد، در حقیقت همان پورت مربوطه در هدر بسته‌های TCP است که این امکان را فراهم می‌کند که از یک دستگاه، چندین ارتباط با کارگزار پیام به وجود آید. سپس این بسته دریافت شده و در مرکز توزیع بررسی می‌شود. می‌دانیم که بسته‌ها در پروتکل MQTT یک تاپیک دارند که گیرنده اطلاعات در مقصد با کمک آن مشخص می‌شود. پس در مرکز توزیع بسته‌ها متناسب با

---

۱ Producer

۲ Consumer

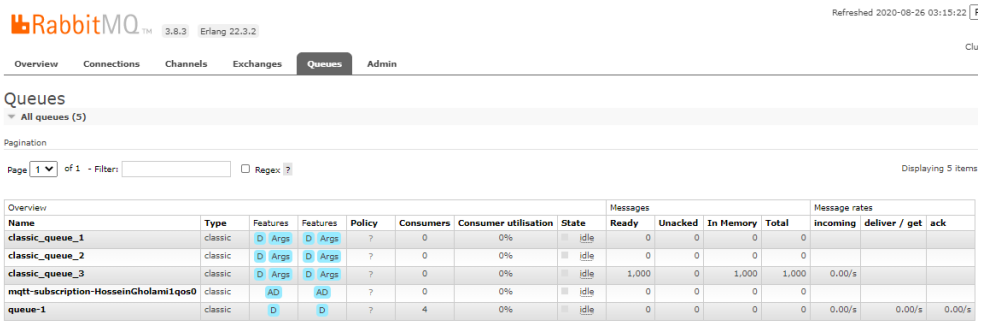
۳ Queue

۴ Router(Exchange)

۵ Channels

تاپیک خود، در صف‌هایی که مربوط به خود هستند، قرار خواهند گرفت. در ادامه نحوه مدیریت مرکز توزیع به‌طور کامل شرح داده می‌شود سپس بسته‌ها در صف‌هایی از پیش تعریف‌شده در نرم‌افزار ذخیره می‌شوند. در این نرم‌افزار می‌توان تنظیماتی دقیق برای هر صف انجام داد و جزئیاتی را در آن پیاده‌سازی کرد که در ادامه شرح داده می‌شود. در پایان نیز یک مصرف‌کننده اطلاعات به این صف‌ها متصل می‌شود که این بسته‌ها را از صف‌ها دریافت کرده و پردازش مربوطه را روی آن انجام می‌دهد. یکی از ویژگی‌هایی که ما را به استفاده از RabbitMQ سوق می‌دهد، این است که در سمت گیرنده اطلاعات دیگر نیازی به به‌کارگیری این پروتکل نیست و می‌توان از پروتکل‌های دیگر که نرم‌افزار پشتیبانی می‌کند، استفاده نمود.

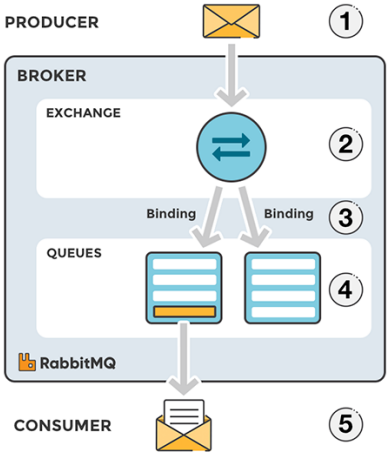
RabbitMQ به ما یک پنل داشبورد برای بررسی شرایط صف‌ها، تبادل داده‌ها و کانال‌ها و ارتباطات می‌دهد که در آن می‌توان اطلاعات مفیدی هنگام به‌کارگیری در عمل به دست آورد. در شکل ۴–۱۰ صفحه مربوط به‌صف‌های را می‌توانید مشاهده کنید، برای مثال در این تصویر ۴ مصرف‌کننده اطلاعات بر روی صف queue-۱ قرارگرفته شده تا در صورت انتقال اطلاعات به این صف، پیام‌ها دریافت شوند. درحالی‌که در صف classic_queue_۳ هزار پیام در حافظه قرارگرفته و هیچ مصرف‌کننده اطلاعاتی به این صف متصل نیست.



شکل ۴–۱۰- داشبورد مدیریتی نرم‌افزار RabbitMQ

با توجه به توضیحات فوق و بررسی وظایف کلی هر بخش، به بررسی دقیق‌تر انتقال پیام‌ها از مرکز توزیع وصف‌ها می‌پردازیم.

در این پروژه برای سمت گیرنده اطلاعات از پروتکل MQTT استفاده شد به این معنا که بسته‌های حاوی اطلاعات سنسوری پس از دریافت توسط این پروتکل، به مرکز انتقال ex.mqtt منتقل می‌شوند. می‌دانیم بسته‌هایی که با این پروتکل ارسال می‌شوند در خود بخشی با عنوان تاپیک دارند، RabbitMQ از این بخش هنگام تقسیم بسته‌ها در صف استفاده کرده و متناسب با مجموعه قوانینی قابل تنظیم بسته‌ها را در صف‌های مربوطه قرار می‌دهد. در شکل ۴–۱۱ نمونه‌ای از این انتقال را می‌بینیم.



شکل ۴-۱۱-انتقال پیام در نرم‌افزار RabbitMQ [۲۳]

هنگامی‌که هر صف تعریف می‌شود، در خود بخشی با عنوان «کلید مسیریابی[۱]» دارد و باید به یک مرکز انتقال متصل شود. صف‌ها با کمک کلید انتقال بسته‌هایی که وارد، یک مرکز انتقال می‌شوند را می‌توانند در خود جای دهند. برای مثال stock.usd.ny و quick.orange.rabit می‌توانند هر دو نام تاپیک برای بسته‌ها باشند و صف‌هایی با کلید مسیریابی فوق می‌توانند، این پیام‌ها را دریافت کنند. دراین‌بین برای RabbitMQ از سه کاراکتر '.' و '#' و'*' برای مدیریت کردن بهتر بسته‌ها هنگام تقسیم بسته‌ها میان صف‌ها استفاده می‌کند.

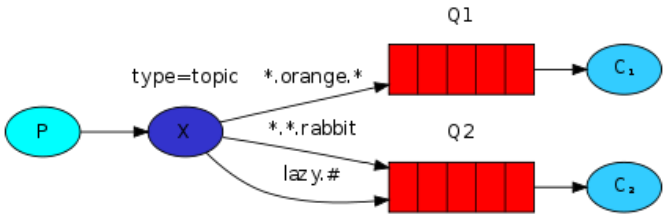
کاراکتر '.' به معنی ایجاد ساختار است و هنگامی‌که در کلید مسیریابی استفاده شود باید دقیقا تاپیک پیام و کلیدمسیریابی یکسان باشد. کاراکتر '*' به این معنی است که در بخش‌هایی که این کاراکتر قرارگرفته می‌تولند تاپیک بسته با کلیدمسیریابی متفاوت باشد و کاراکتر '#' به این معناست، درصورتی‌که تا قبل از این کاراکتر یکسان بود، بسته به صف منتقل شود. مثال زیر را در نظر بگیرید:

بسته‌ای با تاپیک quick.orange.rabbit به یک مرکز انتقال می‌رسد و چهار صف با کلیدهای مسیریابی زیر به این مرکز انتقال متصل هستند.

حالت اول کلیدمسیریابی quick.orange است، این پیام دریافت نمی‌شود، چراکه تعداد بخش‌های ایجادشده با '.' یکسان نیست. حالت دوم کلیدمسیریابی *.orange.* است، این پیام دریافت می‌شود، چراکه تعداد بخش‌های ایجادشده با نقطه یکسان و از دو بخش اول و آخر بسته صرف‌نظر می‌شود. حالت سوم، کلیدمسیریابی quick.# است، این پیام دریافت می‌شود چراکه تاپیک بسته با هر آنچه که قبل از

[۱] Routing key

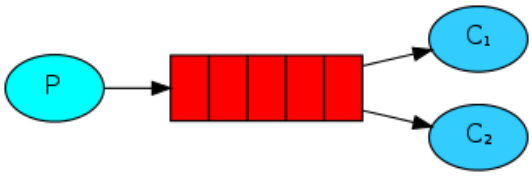
’#‘ است، یکسان می‌باشد. این مکانیزم می‌تواند برای دسته‌بندی پیام‌های ورودی بسیار مؤثر باشد. در شکل ۴–۱۲ می‌توان مثال دیگری را مشاهده نمود.



شکل ۴–۱۲–نحوه استفاده از کلیدمسیریابی برای صف‌ها

مکانیزم نگه‌داری پیام‌ها در صف به‌طورکلی به دو صورت تقسیم‌بندی می‌شود پیام‌های گذرا[۱] و پیام‌های ثابت‌شده[۲]، پیام‌ها، به‌طور عمومی در حافظه نوشته می‌شوند (گذرا هستند) ولی درصورتی‌که حد آستانه‌ای که برای صف تنظیم‌شده بیشتر شوند پیام‌ها به ثابت‌شده، تبدیل شده و روی دیسک نوشته می‌شوند و تنها یک اشاره‌گر که محل ابتدای آدرس دیسک ثبت‌شده است، در حافظه نگهداری می‌شود. هنگامی‌که مصرف‌کننده اطلاعات، درخواست اطلاعات می‌کند، قبل از ارسال این پیام وارد حافظه شده و در حالت گذرا قرار می‌گیرد و پس‌ازاینکه دریافت‌کننده تصدیق انجام کار آن اطلاعات را ارسال کرد از حافظه نیز پاک می‌شوند.

در این ساختار همان‌طور که پیش‌تر بررسی شد، می‌توانیم از چندین سرویس‌دهنده به‌طور هم‌زمان استفاده کنیم تا بتوانیم حجم بسته‌های داخل صف را مدیریت کنیم. در این سیستم یک پارامتری تحت عنوان پیش‌دستی[۳] تعریف می‌شود، به این معنا که تعداد بسته‌های درخواستی از صف را کنترل می‌کند که بسته‌ها به‌صورت چندتایی ارسال شوند. در این حالت، بسته‌ها به حالت گذرا تغییر وضعیت یافته تا هنگامی‌که سرویس‌دهی این بسته‌ها تمام شد، یک تعداد بسته جدید تقاضا شود. این امر موجب می‌شود تعداد پیام‌هایی که برای ارسال و دریافت بین سرور صف و سرویس‌دهنده منتقل می‌شود، کاهش یابد و درنتیجه بازدهی بالاتر رود. برای مثال به شکل ۴–۱۳ توجه کنید.



شکل ۴–۱۳–اتصال دو مصرف‌کننده‌ی اطلاعات به یک صف

---

۱ Transient

۲ Persistent

۳ Prefetch

فرض کنید در این شکل دو مصرف‌کننده‌ی اطلاعات، به این صف متصل شده‌اند و در صف ۵۰ بسته وجود دارد. C۱ با پیش‌دستی ۵ به این صف متصل شده و C۲ که کامپیوتری با امکانات سخت‌افزاری قوی‌تری است، با پیش‌دستی ۱۰ به این صف متصل می‌شود. حال در لحظه صفر C۱ درخواست ۵ بسته و C۲ درخواست ۱۰ بسته می‌کند. در این حالت همچنان ۵۰ بسته در صف وجود دارد، با این تفاوت که ۱۵ بسته به حالت گذرا درآمده است. بعد از پردازش توسط C۱ و C۲ اعلان پایان کار داده و تقاضای بسته جدید کرده به این شکل بسته‌ها از صف حذف‌شده و بسته‌های جدید برای آن‌ها ارسال می‌شود.

۴-۳-۱ راه‌اندازی RabbitMQ با داکر

از پیش‌تر در بخش مروری بر داکر به یاد داریم که می‌توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد. RabbitMQ خود یکی از ایمیج های استاندارد است که به‌سادگی از آن استفاده می‌شود. برای این کار می‌توانیم در داخل داکر فایل از ایمیج استاندارد شروع کرده و پلاگین‌هایی که قصد فعال‌سازی آن را داریم، با اجرای دستور مربوطه به‌کارگیریم تا ایمیج جدیدی با ویژگی‌هایی که مدنظر داریم ساخته شود. در شکل ۴–۱۴ محتوای داکر فایل را می‌توان مشاهده کرد.

```
FROM rabbitmq:3-management
RUN rabbitmq-plugins enable rabbitmq_management
RUN rabbitmq-plugins enable  rabbitmq_mqtt -offline
```

شکل ۴–۱۴-محتوای داکر فایل برای راه‌اندازی RabbitMQ

حال با به‌کارگیری داکر کامپوز تنها کافی است دستور دهیم که ایمیج فوق ساخته شود و پورت مخصوص به ارتباط با کانتینر و کامپیوتر میزبان برقرار شود. همچنین می‌توان فایل تنظیمات را تغییر داد و آن را در داخل کانتینر قرار داد. در شکل ۴–۱۵ می‌توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر RabbitMQ نوشته‌شده را مشاهده کرد.

```
queue:
   build: rbmq/.
   container_name: rabbitmq
   ports:
     - 5672:5672
     - 15672:15672
     - 1884:1884
   volumes:
     - "./rbmq/Conf/:/etc/rabbitmq/"
```

شکل ۴–۱۵- بخش مرتبط با RabbitMQ در داکر کامپوز

٤-٤ مروری بر ذخیره کننده Redis

Redis یک ذخیره کننده ساختار اطلاعات، داخل حافظه‌ای متن‌باز است که می‌تواند به‌عنوان پایگاه‌داده، حافظه نهان و ابزاری برای یکپارچه‌سازی عمل کند. Redis از ساختارهای اطلاعات متفاوتی ازجمله؛ رشته‌ها، هش‌ها، لیست‌ها، مجموعه‌ها، مجموعه‌های مرتب‌شده برای کاربردهای متفاوت و جریان‌های اطلاعات (سری زمانی‌ها) پشتیبانی می‌کند و توانایی اجرا بر روی خوشه‌ای از کامپیوترها را دارا است که بزرگ‌ترین مزیت استفاده از Redis است.

در این نرم‌افزار بعضی از امکانات محاسباتی ازجمله اضافه کردن به رشته‌ها، تغییر در اعداد، تغییر در لیست‌ها، محاسبه اشتراکات دو مجموعه، بررسی المان‌های منحصربه‌فرد و ...را انجام داد.Redis به زبان ANSI C نوشته‌شده و بر روی همه‌ی سیستم‌عامل‌ها بدون هیچ‌گونه پیش‌نیاز قابل‌اجرا است. اگرچه این برنامه بر روی همه‌ی سیستم‌عامل‌ها توانایی اجرا دارد ولی خود برنامه، استفاده از سری سیستم‌عامل‌های Linux را پیشنهاد می‌دهد.

درست است که Redis اطلاعات را در حافظه ذخیره می‌کند اما به این معنی نیست که پس از خاموش شدن و یا هر اتفاقی که باعث خالی شدن حافظه شود، داده‌های ما پاک می‌شوند. بلکه Redis برای نگه‌داری دائمی داده‌ها آن‌ها را با توجه به تنظیماتی که برای آن مشخص کرده‌ایم به دیسک اصلی سیستم منتقل کرده و بعد از پاک شدن حافظه مجدد می‌تواند آن‌ها را منتقل کند و کار را از سر بگیرد. این ویژگی باعث شده اصطلاحاً به آن on-disk persistence بگویند.

زمانی از ذخیره‌سازی موقت[1] استفاده می‌شود که قصد داشته باشیم دسترسی به هارددیسک کمتر انجام شود. به‌عبارت‌دیگر در ذخیره‌سازی موقت، اطلاعات در حافظه موقت ذخیره می‌شود که این فرآیند سرعت دسترسی به اطلاعات و بارگذاری آن‌ها را افزایش می‌دهد. از این طریق در کنار صرفه‌جویی در زمان و افزایش سرعت، دسترسی کمتری به منابع موردنیاز انجام می‌شود. این امر نیز به بهینه‌سازی بیشتر کمک می‌کند. به این نکته نیز باید اشاره کرد که در Redis اطلاعات در حافظه ذخیره می‌شوند، این امر باعث می‌شود دسترسی به آن‌ها با سرعت بسیار بیشتری انجام شود؛ اما این سکه روی دیگری نیز دارد و امکان ذخیره‌سازی دائمی اطلاعات را در Redis وجود نخواهد داشت. در این پژوهش ازآنجاکه پس از تجمیع اطلاعات نیازمند پردازش این اطلاعات هستیم تا به رانندگی افراد امتیازی اختصاص دهیم از این نرم‌افزار استفاده شد.

[1] Caching

درRedis دو عنصر، کلید و مقدار[۱] داریم. عنصر مقدار می‌تواند انواع مختلفی داشته باشد. اجازه دهید به برخی این مقادیر و کاربردهای آن‌ها نگاهی بیندازیم.

رشته‌ها: اگر مقدار از نوع رشته بود، می‌توان عملیات درج، بهنگام سازی، حذف، دریافت را از آن کلید انجام داد. مثلاً با دستور زیر می‌توان یک کلید ساخت و یک‌رشته در آن درج کرد:

یک کلید به اسم user_۱ ساخته‌شده و book۱ به‌عنوان مقدار برای این کلید در نظر گرفته می‌شود.	SET user_۱ book۱
مقدار مورد (book۱) نظر برای کلید user_۱ را برگردانده می‌شود.	GET user_۱

لیست: با استفاده از لیست می‌توان یک آرایه دلخواه داشت که بتوان در این آرایه، عنصری را اضافه یا کم کرد و عملیات مختلف دیگر را انجام داد.

دسته‌ها[۲]: اگر مقدار موجود از نوع دسته‌ها باشد درواقع یک لیست وجود دارد که هیچ‌کدام از عناصر آن تکراری نیستند.

دسته‌های منظم شده[۳]: همانند مقدارهایی از نوع دسته است با این تفاوت که هر عنصر از مجموعه دارای وزن است و این وزن می‌تواند به‌صورت مرتب نگهداری شود. (مثلاً از وزن کم به زیاد)

هش‌ها[۴]: اگر با انواع JSON آشنایی داشته باشید درک مقادیر هش‌ها ساده‌تر است. این مقادیر می‌توانند اشیایی مانند JSON را در خود جای دهند. بدین شکل که ابتدا کلیدی برای ذخیره‌سازی اطلاعات دریافت کرده و سپس به ذخیره‌سازی اطلاعات به‌صورت زوج مرتب‌هایی از کلید و مقدار می‌پردازد.

علاوه بر موارد فوق، Redis می‌تواند مقادیر دیگری را نیز ذخیره و بازیابی کند. یکی از انواع آن‌ها، HyperLogLog است. فرض کنید می‌خواهید تعداد تکرار یک عنصر خاص از یک لیست را به دست آورید. اگر این لیست بسیار بزرگ باشد، این کار به‌راحتی انجام نمی‌پذیرد.

---

۱ Value

۲ Sets

۳ Sorted Sets

۴ Hashes

مقادیر HyperLogLog می‌تواند با دقت بسیار بالا (اما نه ۱۰۰ درصد) تعداد تکرار یک عنصر خاص را حدس بزند. این کار با استفاده از الگوریتم‌های خاصی امکان‌پذیر است. انواع دیگری مانند ذخیره‌سازی عناصری از جنس موقعیت‌های مکانی و یا Bitmap های نیز در Redis وجود دارند که کاربردهای خاص خود را دارند.

در این پژوهش ازآنجاکه میان رانندگان و اطلاعات ارسالی آن‌ها تمایز قائل هستیم و تنها قصد انتقال رویدادهای رانندگی و برچسب زمانی آن‌ها را داریم، می‌توانیم اطلاعات خود را به‌صورت هش‌ها در Redis درآورده و به این شکل ذخیره‌سازی اطلاعات را پیاده‌سازی کنیم. اطلاعات با دستور و ساختار زیر ذخیره‌شده‌اند.

کلید ذخیره‌سازی اطلاعات بر اساس شماره شناسایی راننده تعیین می‌شود. ازآنجایی‌که رشته‌های هر کلید باید نسبت به یکدیگر منحصربه‌فرد باشند، از برچسب زمان به‌عنوان رشته و در بخش مقدار آن رویداد رانندگی موردنظر نوشته می‌شود. با دستور HSET می‌توان یک سری اطلاعات هش ساخت و اطلاعات رشته و مقدار را به آن، مانند لیست، اضافه کرد و با دستور HGETALL می‌توان اطلاعات ذخیره‌شده در یک هش را به دست آورد. در شکل ۴–۱۶ نمونه‌ای از استفاده از این دستورات و بارگذاری این اطلاعات در سرویس‌دهنده Redis می‌توانیم مشاهده کنیم.

```
127.0.0.1:6379> Hset Driver1 "14/05/2016 11:17:09" "troca_faixa_direita_agressiva"
(integer) 1
127.0.0.1:6379> Hset Driver1 "14/05/2016 11:21:12" "freada_agressiva"
(integer) 1
127.0.0.1:6379> Hset Driver1 "14/05/2016 11:21:20" "freada_agressiva"
(integer) 1
127.0.0.1:6379> Hset Driver1 "14/05/2016 11:22:21" "aceleracao_agressiva"
(integer) 1
127.0.0.1:6379> HGETALL Driver1
1) "14/05/2016 11:17:09"
2) "troca_faixa_direita_agressiva"
3) "14/05/2016 11:21:12"
4) "freada_agressiva"
5) "14/05/2016 11:21:20"
6) "freada_agressiva"
7) "14/05/2016 11:22:21"
8) "aceleracao_agressiva"
```

شکل ۴–۱۶- بررسی دستورات کار کردن با هش‌ها در Redis

در شکل فوق با برنامه redis-cli به سرور Redis متصل شدیم و سعی کردیم دستورات را در خط دستور وارد کنیم. در ادامه با یک برنامه پایتون، سعی می‌شود اطلاعات را از نرم‌افزار RabbitMQ خوانده و به سرویس‌دهنده Redis منتقل کرد.

۴-۴-۱ راه‌اندازی Redis با داکر کامپوز

از پیش‌تر در بخش مروری بر داکر به یاد داریم که می‌توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد، Redis یکی از ایمیج های استاندارد است که می‌توان به‌سادگی از آن استفاده نمود، برای استفاده از این ایمیج با داکر کامپوز، تنها کافی است، پورت مخصوص به ارتباط با کانتینر آن را به کامپیوتر میزبان متصل کرده و فضایی در کامپیوتر میزبان برای ذخیره‌سازی اطلاعات برای آن اختصاص دهیم تا با شروع به کار مجدد کانتینر، اطلاعات ذخیره‌شده در آن از بین نرود، همچنین می‌توان فایل تنظیمات مربوط به Redis را تغییر داد و در کانتینر قرارداد و دستور داد تا برنامه Redis بر اساس تنظیماتی که در کانتینر قرارگرفته، اجرا شود. در شکل ۴–۱۷ می‌توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر Redis نوشته‌شده را مشاهده کرد.

```
cache:
    image: redis:latest
    container_name: redis
    ports:
        - 6379:6379
    volumes:
        - ./redis/config/redis.conf:/redis.conf
        - ./redis/Data:/data
    command: [ “redis-server”, “/redis.conf” ]
```

شکل ۴–۱۷- بخش مرتبط با Redis در داکر کامپوز

۴-۵ پیاده‌سازی سیستم دریافت و ذخیره‌سازی

در این بخش قصد داریم تا به بررسی کلی سیستم دریافت و ذخیره‌سازی اطلاعات بپردازیم، در بخش‌های قبل با نرم‌افزارهای مورداستفاده آشنا شدیم و راه‌اندازی آن‌ها با کمک داکر را بررسی کردیم. حال به بررسی عملکرد هرکدام از نرم‌افزارهای فوق در معماری مورداستفاده به‌صورت جدا پرداخته و ارتباط آن‌ها با یکدیگر را معرفی می‌کنیم.

همان‌طور که پیش‌تر در شکل ۴–۸ مشاهده کردیم، برای پیاده‌سازی این سیستم از سه برنامه مجزا استفاده شد که با یکدیگر در تعامل هستند. برنامه‌هایی به شرح: صف[۱] مصرف‌کننده[۲] ذخیره موقت[۳].

برنامه صف با کمک RabbitMQ پیاده‌سازی شد. بدین شکل که خودروها با پروتکل MQTT بر روی پورت ۱۸۸۴ با شناسه هویتی مشخص، به این نرم‌افزار متصل و بسته‌های خود را با تاپیک مشخص (شماره خودروی خود) به مرکز انتقال "ex.mqtt" منتقل می‌کنند. در ادامه همان‌طور که در شکل ۴–۱۸ می‌بینید، بسته‌ها با کلیدهای مسیریابی متفاوت وارد صف‌های مربوط به خود می‌شوند. برای مثال بسته‌ای با تاپیک Car.۳۲۱ بر اساس کلیدمسیریابی Car.* به‌صف classic_queue_۱ منتقل می‌شود و در صف نگهداری می‌شود تا یک مصرف‌کننده اطلاعات را از صف بخواند.

شکل ۴–۱۸- فرایند انتقال بسته‌ها از بخش توزیع دادگان به صف‌ها

در ادامه می‌بایست نرم‌افزاری طراحی کنیم که اطلاعات را از صف‌های RabbitMQ خوانده و به Redis منتقل کند. تا فرایند بعدی که امتیاز دادن به رانندگی شخص در هر سفر است، انجام شود.

می‌دانیم می‌توان بسته‌های قرارگرفته داخل صف را بر روی پورت ۵۶۷۲ و پروتکل AMQP-۰-۹۱ با شناسه هویتی مشخص توسط کتابخانه pika که در پایتون نوشته‌شده، دریافت کنیم. برای انجام این کار به‌صورت بهینه باید حجم بسته‌های درون صف RabbitMQ از حدی بیشتر نشود. مکانیزم دریافت اطلاعات از صف به‌صورت موازی و چند رشته پیاده‌سازی شد. بدین شکل که با کمک مکانیزم

[۱] Queue

[۲] Consumer

[۳] Cache

برنامه‌نویسی چند رشته‌ای و کتابخانه threading، می‌توان به تعداد دلخواه، مصرف‌کننده اطلاعات ساخت که به صف‌ها متصل شوند. کلاس مربوط به این مدل را می‌توان در شکل ۴–۱۹ مشاهده کرد.

```
class rbmq(threading.Thread):
    # Thread class with a _stop() method.
    def __init__(self,Slug,Parameter,prefetch_count,QueueName,CalbackFunc,*args, **kwargs):
        super(rbmq, self).__init__(*args, **kwargs)
        self.slug=Slug
        self.parameters=Parameter
        self.pckh=prefetch_count
        self.queuename=QueueName
        self.callbackfunc=CalbackFunc
    def run(self):
        # target function of the thread class
        try:
            connection = pika.BlockingConnection(self.parameters)
            channel = connection.channel()
            channel.basic_qos(prefetch_count=self.pckh)
            channel.basic_consume(queue=self.queuename,
                          on_message_callback=self.callbackfunc,
                          consumer_tag=self.slug)
            print( self.slug ,' [*] Waiting for messages on :',self.queuename )
            channel.start_consuming()
        finally:
            connection.close()
            print(self.slug ,'  Stoped from ',self.queuename )
    def get_id(self):
        # returns id of the respective thread
        if hasattr(self, '_thread_id'):
            return self._thread_id
        for id, thread in threading._active.items():
            if thread is self:
                return id
    def stop(self):
        thread_id = self.get_id()
        res = ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id,
              ctypes.py_object(SystemExit))
        if res > 1:
            ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id, 0)
            print('Exception raise failure')
```

شکل ۴–۱۹-کلاس مربوطه به برنامه انتقال‌دهنده اطلاعات RabbitMQ به Redis

در ادامه وظیفه‌ای که هرکدام از این رشته‌ها دارند، هنگام دریافت پیام تعیین می‌شود که با پیام دریافت شده، چه عملی را انجام دهند. برای انجام این کار هنگامی‌که قصد داریم شئ ای از کلاس فوق بسازیم تابعی را به‌عنوان CallbackFunc به آن معرفی می‌کنیم تا وظایفی که هنگام دریافت بسته‌ها باید انجام شود را تعریف کنیم. هنگامی‌که بسته‌ها از صف‌ها دریافت شدند، باید به نرم‌افزار ذخیره‌سازی موقت (Redis) منتقل شوند. در بخش قبل بررسی کردیم که برای این کار از مکانیزم هش‌ها در Redis استفاده خواهیم کرد. به همین منظور با استفاده از کتابخانه Redis در پایتون، به‌صورت کاربر Redis به این سرویس‌دهنده متصل شده و اطلاعات خارج‌شده از صف را در این نرم‌افزار قرار می‌دهیم.

در شـــکل ۴–۲۰ تابعی که وظیفه انتقال اطلاعات به ســـرویس‌دهنده Redis را دارد مشـــاهده می‌کنیم، درصـــورتی‌که این برنامه نتواند اطلاعات را به Redis منتقل کند، اطلاعات را در بخشـــی از کانتینر اجراکننده این برنامه که به کامپیوتر میزبان متصـل شـده، منتقل می‌کند. اگر روند اجرای برنامه مشـکلی نداشت، پیام تصدیق برای برنامه صف ارسال می‌شود تا پیام را از حافظه خود پاک کند.

```
def Packet_Handeler_callback(ch, method, properties, body):
    Str=str(body.decode("utf-8"))
    Data=json.loads(Str)
    client = redis.Redis(host='redis', port=6379)
    send_correct = client.hset(name=Data['driver_id'],key=Data['timestamp'],value=Data['event'])
    # if Fails , save on log file
    if not(send_correct):
        with open(".//log//log.txt", "a") as myfile :
            myfile.write(Str)
    ch.basic_ack(delivery_tag = method.delivery_tag)
```

شکل ۴–۲۰-تابع فراخوانی اطلاعات هنگام دریافت اطلاعات از Redis و انتقال به RabbitMQ

٤-٦ جمع بندی

در بخش‌های فوق هر یک از نرم‌افزارهای به‌کاررفته برای دریافت و ذخیره‌سازی اطلاعات بررسی کردیم و در پایان برنامه‌ای که وظیفه یکپارچه‌ســازی این نرم‌افزارها را دارد، مرور کردیم، حال وقت آن رســیده که به بررســی اطلاعاتی که در ســرویس‌دهنده Redis جمع شــده بپردازیم. تا رانندگی افراد را مورد ارزیابی قرار دهیم.

۵ فصل پنجم

مدل ارزیابی‌کننده اطلاعات رانندگان

مدل ارزیابی‌کننده اطلاعات رانندگان

در بخش‌های قبل شیوه تشخیص و برچسب‌گذاری اطلاعات سنسوری را بررسی کردیم و به نحوه تجمیع این اطلاعات پرداختیم. حال در این بخش به دنبال راهکاری هستیم که بتوانیم رانندگی افراد را بررسی و نمره دهی کنیم. اولین ایده‌ای که به ذهن می‌رسد، مقایسه رانندگان با یکدیگر است. برای این کار به یک جامعه آماری بزرگ‌تر نیاز داریم تا بتوان مقایسه‌ای عادلانه انجام گیرد. مجموعه دادگانی که پیش‌تر برای تشخیص رویدادهای رانندگی مورداستفاده قراردادیم، حجم اطلاعات زیادی نداشت پس باید به سراغ مجموعه دادگان بزرگ‌تری برویم. در این بخش ابتدا مجموعه دادگان جدید که شامل رویدادهای رانندگی افراد است را بررسی کرده و سپس سعی می‌کنیم مکانیزمی برای نمره‌دهی رانندگان طراحی کنیم.

۵-۱ جمع‌آوری اطلاعات و آنالیز اکتشافی

مجموعه دادگان بکار رفته در این بخش از همکاری دو شرکت ماکروسافت[۱] و شرکت پوینترتلوکیشن[۲] به دست آمد. [۲٤] پوینترتلوکیشن، یک شرکت توسعه دستگاه‌های مخابراتی برای خودروها است که روی مدیریت ناوگان حمل‌ونقل تمرکز کرده است. دستگاه‌های طراحی‌شده توسط این شرکت ویژگی‌های خودرو همچون سرعت، مکان و همچنین برخی از رفتارهای راننده خودروها همچون چرخش‌های خطرناک، ترمزگیری خطرناک، سرعت‌گیری خطرناک را می‌تواند تشخیص دهد. این شرکت در طی یک همکاری با ماکروسافت اطلاعات تعداد زیادی از رانندگان خود را منتشر کرد تا بتواند مسئله مقایسه رانندگان را حل کند، ماکروسافت پس از ارائه راه‌حل این مسئله، مجموعه دادگان را در اختیار عموم قرار داد. در این پروژه با به‌کارگیری همان مجموعه اطلاعات و تنها با تغییر در برخی از بردارهای ویژگی به نتایج جالبی دست‌یافتیم که در ادامه بررسی خواهیم نمود.

این مجموعه دادگان حاوی اطلاعات متفاوتی است ولی به‌طورکلی اطلاعات را می‌توان به دو نوع تقسیم نمود؛ اطلاعات رفتاری و اطلاعات عملکردی، همان‌طور که در شکل ۵–۱ مشاهده می‌کنید در ستون EventName، نوع رویدادهایی که به سمت سرور منتقل‌شده نشان داده‌شده است. این ستون رویدادهای رفتاری راننده را در اختیار ما قرار می‌دهند و در سایر ستون‌ها اطلاعات عملکردی هر خودرو مشخص‌شده، به‌طور مثال در ستون DriverId راننده‌ای که پشت سیستم است معرفی‌شده و همچنین

---

[۱] Microsoft : www.Microsoft.com

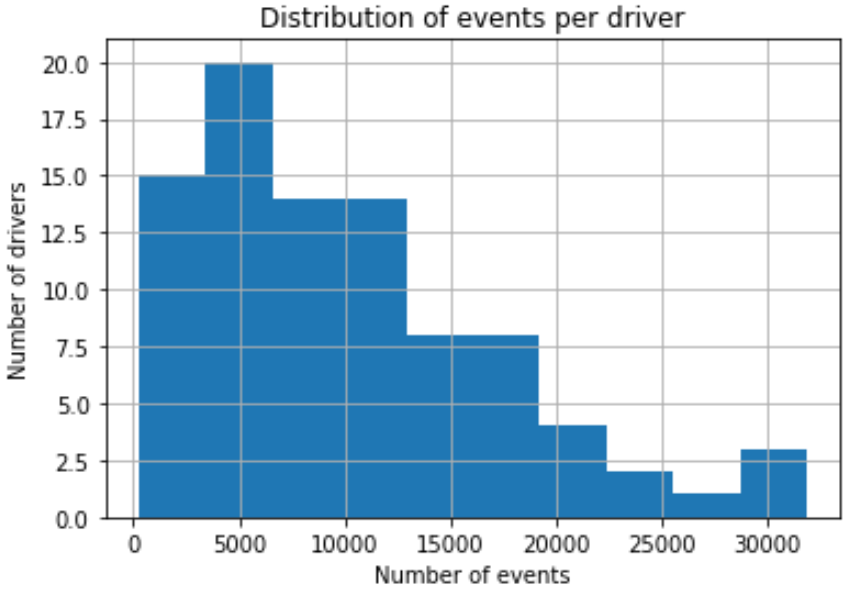
[۲] Pointer Telocation : www.pointer.com

اطلاعاتی که از حسگر GPS به دست می‌آید همچون طول و عرض جغرافیایی و سرعت خودرو برچسب زمانی خورده‌اند.

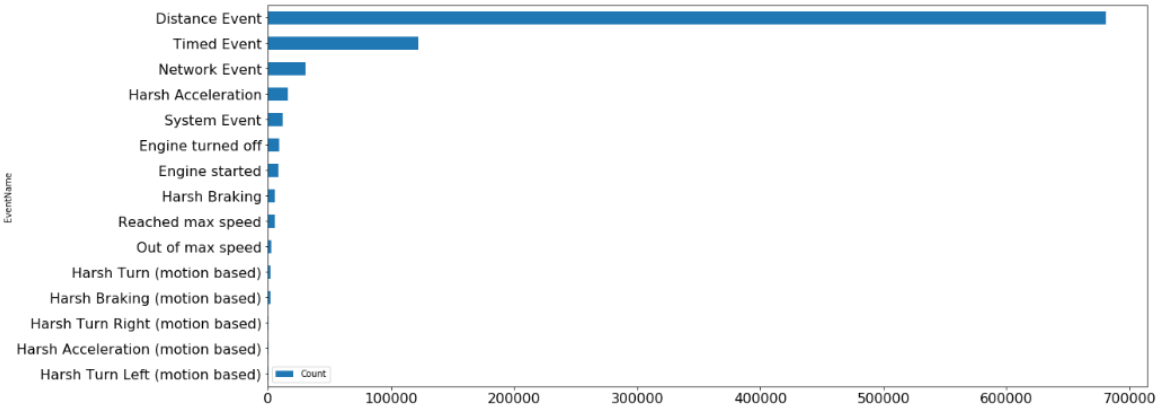
DriverId	EventName	Latitude	Longitude	Speed km/h	ts
0	Distance Event	34.1918	-118.1112	51.0000	2017-11-02 15:33:06.010
0	Distance Event	34.1921	-118.1122	37.0000	2017-11-02 15:33:14.100
0	Harsh Acceleration	34.1924	-118.1128	52.0000	2017-11-02 15:33:36.500
0	Distance Event	34.1924	-118.1130	60.0000	2017-11-02 15:33:37.650
0	Harsh Acceleration	34.1924	-118.1130	60.0000	2017-11-02 15:33:37.440
0	Harsh Acceleration	34.1923	-118.1131	67.0000	2017-11-02 15:33:38.490
0	Distance Event	34.1924	-118.1130	60.0000	2017-11-02 15:33:37.190
0	Distance Event	34.1920	-118.1140	78.0000	2017-11-02 15:33:42.640
0	Distance Event	34.1914	-118.1150	87.0000	2017-11-02 15:33:47.230
0	Timed Event	34.1914	-118.1150	87.0000	2017-11-02 15:33:48.270

شکل ۵–۱-نمونه‌ای از رویدادهای رفتاری و عملکردی مجموعه دادگان دوم

تعداد کل رانندگان در این مجموعه دادگان ۸۹ نفر است و میانگین تعداد رویدادها برای هر راننده ۷۴۳۵ نمونه است که با توزیع زیر پخش‌شده است؛ اما برای تخمین نوع رانندگی راننده، ما تنها به تعداد رویداد زیاد نیاز نداریم، بلکه به تنوع در نوع رخدادهای موجود نیاز داریم. تا مدل خوبی برای نمره‌دهی و مقایسه رانندگان در اختیار داشته باشیم برای این کار در شکل ۵–۲ می‌توانیم توزیع به ازای رویدادهای متفاوت را بررسی کنیم.



شکل ۵–۲-توزیع تعداد رانندگان بر اساس تعداد رویدادهای تولیدشده

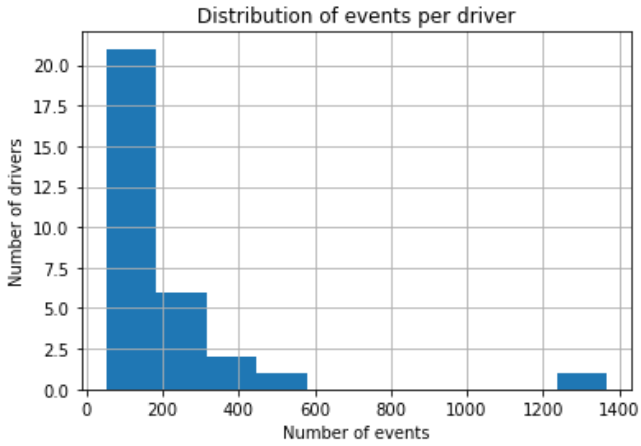


شکل ۵–۳- نمودار تعداد رویداد تولیدشده توسط رانندگان

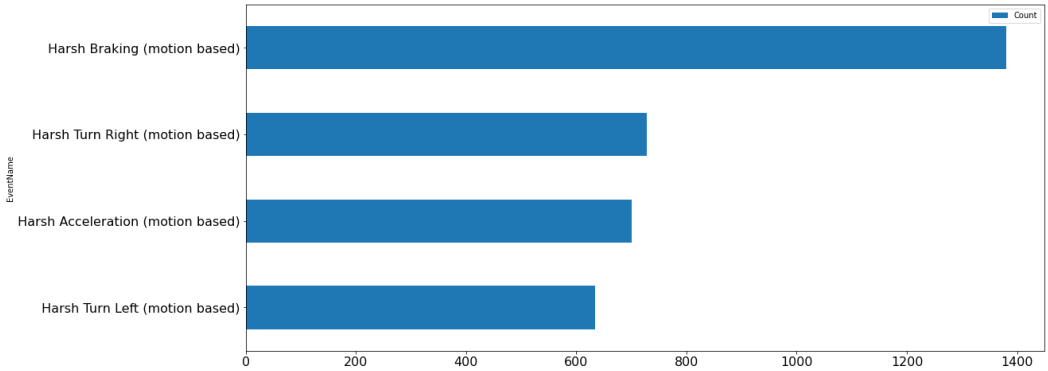
همان‌طور که در شکل ۵–۳ قابل‌مشاهده است، تعدادی از این رویدادها، رویدادهای حرکتی، زمانی، سیستمی و رویدادهای مربوط به اتصالات شبکه است که در روند مقایسه و نمره دهی به عملکرد رانندگان بی‌تأثیر است. همچنین با توجه به تعریف پروژه که پیش‌تر داشتیم ما دسترسی به اطلاعات عملکردی حسگر GPS نداریم. به این معنا که نمی‌توانیم رخدادهای مربوط به‌سرعت غیرمجاز و رسیدن به آستانه حرکت غیرمجاز را بررسی کنیم و تنها می‌توانیم رخدادهای رفتاری که با حسگرهای شتاب‌سنج و ژیروسکوپ به دست می‌آیند را بکار گیریم. همچنین اگر با دقت بیشتری به اطلاعات نگاه بیاندازیم، برای رویدادهای شتابگیری خطرناک و ترمزگیری خطرناک، دو نوع برچسب مشاهده می‌کنیم. در جلوی بعضی از آن‌ها داخل پرانتز کلمه‌ی «بر اساس تغییرات حرکتی[۱]» علامت خورده است. این برچسب به این معنا است که این اطلاعات از حسگرهای موردنظر به‌دست‌آمده ولی سایر آن‌ها بر اساس تغییر حرکت ناگهانی سرعت به‌دست‌آمده پس این ردیف از اطلاعات را می‌بایست حذف کنیم چراکه توسط حسگرهای مدنظر ما به دست نیامده است. پس با توجه به توضیح پیش، باید تعدادی از اطلاعات مربوط را حذف کنیم تا هدف پژوهش خود را ارضا کنیم. به همین منظور در شکل ۵–۴، شکل ۵–۵، می‌توانیم به ترتیب توزیع تعداد رانندگان بر اساس تعداد رویداد تولیدشده و نمودار تعداد رویداد تولیدشده توسط رانندگان را پس از حذف اطلاعات یاد شده مشاهده می‌کنیم.

متأسفانه این مجموعه دادگان معیاری برای تعویض لاین خطرناک در اختیار ما قرار نمی‌دهد، پس‌ازاین رخداد نتوانستیم معیاری جهت ارزیابی رانندگان استفاده کنیم. ولی درصورتی‌که سیستم پیشنهادی بدون این ویژگی به مرحله پیاده‌سازی برسد، با تجمیع اطلاعات می‌توان فرایند آموزش که در ادامه بررسی می‌کنیم را برای این بردار ویژگی نیز انجام داد.

[۱] Motion based



شکل ۵–۴- توزیع تعداد رانندگان بر اساس تعداد رویدادهای تولیدشده پس از حذف اطلاعات نامربوط



شکل ۵–۵- نمودار تعداد رویداد تولیدشده توسط رانندگان پس از حذف اطلاعات نامربوط

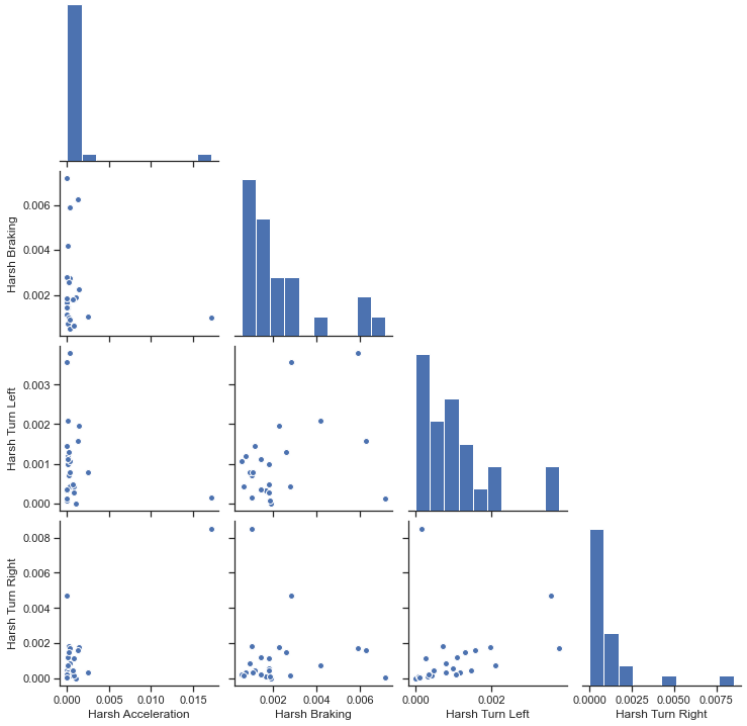
۵-۲ مهندسی بردار ویژگی

ازآنجایی‌که زمان رانندگی رانندگان یکسان نیست، برای یک مقایسه عادلانه می‌بایست زمان رانندگی هر شخص را حساب کنیم و از آن به‌عنوان معیاری بهره بگیریم تا بتوانیم مقایسه‌ای عادلانه انجام دهیم. به همین منظور یک بردار ویژگی برای هر راننده بر اساس تعداد رویداد ساخته‌شده در زمان رانندگی محاسبه کنیم تا مقایسه‌ای عادلانه داشته باشیم و زمان رانندگی را به‌عنوان یک پارامتر به نتیجه تحلیل بیفزاییم، خروجی این فرایند را می‌توان در شکل ۵–۶ مشاهده نمود.

DriverId	F_Harsh Acceleration (motion based)	F_Harsh Braking (motion based)	F_Harsh Turn Left (motion based)	F_Harsh Turn Right (motion based)
0	0.0000	0.0018	0.0010	0.0008
4	0.0000	0.0011	0.0015	0.0004
5	0.0002	0.0010	0.0007	0.0019
6	0.0000	0.0007	0.0012	0.0003
8	0.0013	0.0022	0.0020	0.0018
14	0.0000	0.0017	0.0003	0.0001

شکل ۵–۶-بردار ویژگی هر راننده بر اساس زمان طی شده و تعداد رویداد تولید کرده

در شکل ۵–۷ می‌توان تعداد و توزیع این رویدادها برای همه‌ی اطلاعات مشاهده کرد. ولی همان‌طور که مشخص است در بعضی از نقاط، دادگان پرت و خارج از عرف داریم. پس‌نیاز داریم با کمک روشی بتوانیم این اطلاعات را حذف کنیم. چراکه وقتی حالتی خاص را در مقایسه وارد کنیم ممکن است نتیجه مقایسه را به‌صورت عمومی تحت شعاع خود قرار دهد. برای این کار می‌بایست از روش‌های حذف دادگان پرت استفاده کنیم، به همین منظور از تبدیل BoxCox و قاعده‌ی حذف اطلاعات بالای میانگین و انحراف معیار استفاده شد، در ادامه این روش را توضیح داده و سپس از آن بهره‌گیری می‌کنیم.



شکل ۵–۷- نمودار تعداد و توزیع این رویدادها نسبت به یکدیگر

تبدیل Box-Cox [۲۵]، یک تبدیل آماری قابل بازگشت است که در ۱۹۶۴ توسط دو دانشمند با نام‌های George Box و Sir David Roxbee Cox طراحی‌شده که سعی می‌کند که توزیع داده‌شده را به توزیع نرمال نزدیک کند. این تبدیل به فرم رابطه ۵-۱ تعریف‌شده که از طریق یافتن پارامتر λ مناسب این تبدیل قابل بازگشت است و پارامتر λ از بهینه‌سازی یک تابع خطا که راه‌حل بسته دارد، به دست می‌آید. این تبدیل در پایتون، در کتابخانه‌ی scipy.stats.boxcox پیاده‌سازی شده، همچنین تبدیل معکوس آن را نیز می‌توان در scipy.special.inv_boxcox یافت. این تبدیل به فرم ریاضی به شکل زیر تعریف می‌شود:

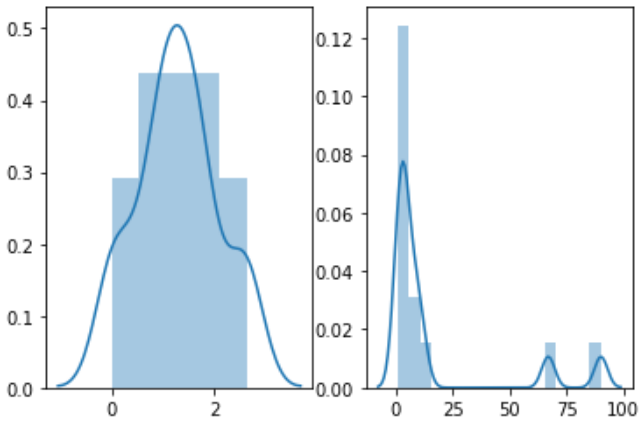
$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda} - ۱}{\lambda} & if\ \lambda \neq ۰, \\ \ln y_i & if\ \lambda = ۰, \end{cases}$$

رابطه (۵–۱)

به‌عنوان‌مثال رشته اعداد x را به X تبدیل می‌کند.

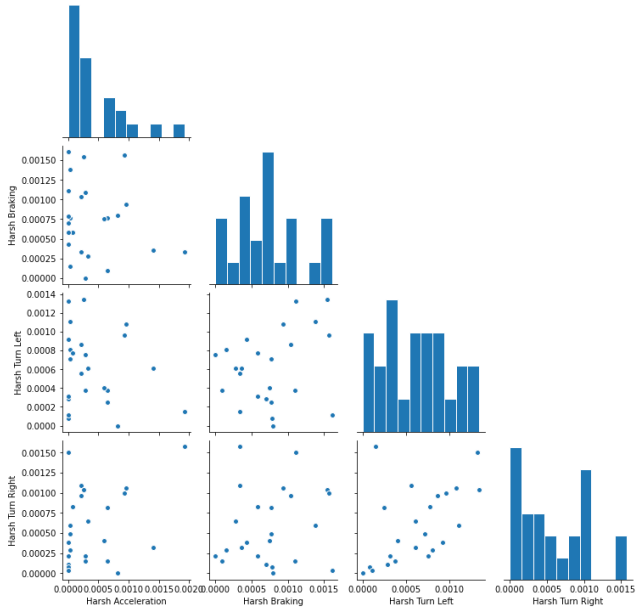
```
x [ 1.1,12,2,3,4,5,67,9,9,5,1,2.9,90.1]

X [0.0941243,1.82374859,0.63357865,0.95400214,1.16147145,1.31191724,
   2.5427535,1.66838902,1.66838902,1.31191724,0,0.92850195,2.63694971]
```

با انجام این تبدیل، پارامتر λ معادل ۰.۲۶۳۲۷۸۷۲۳۸۹۸۵۹۴۳۶- محاسبه می‌شود و درصورتی‌که با همین پارامتر این اعداد را تبدیل کنیم به اعداد قبلی می‌رسیم. در شکل ۵–۸ نمودار توزیع این دو رشته را مشاهده می‌کنیم. در شکل ۵–۸-راست، به‌سادگی قابل‌مشاهده است که نمونه‌های ۶۷ و ۹۰.۱ جزو دادگان پرت محسوب می‌شوند. ولی در شکل ۵–۸-چپ این اعداد همگی در یک توزیع شبیه به نرمال قرارگرفته‌اند.



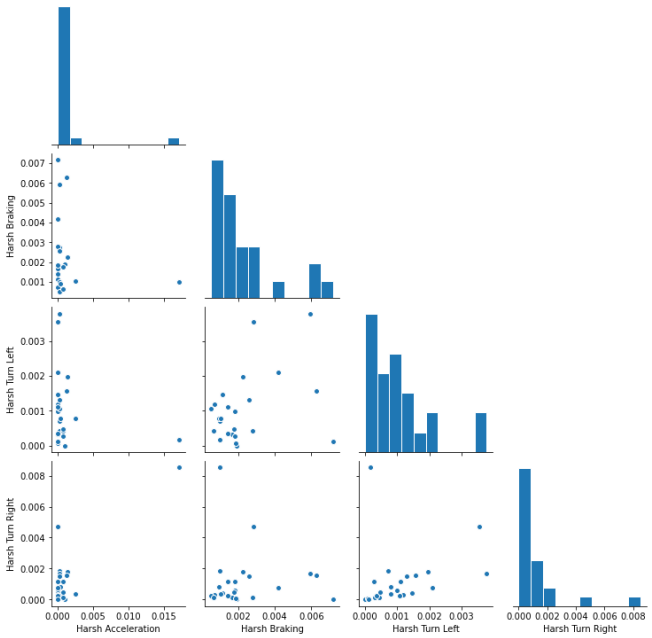
شکل ۵–۸- نمودار نمونه تبدیل box-cox برای بردار x

حال که شیوه محاسبه این تبدیل را فراگرفتیم وقت آن رسیده که آن را بر روی اطلاعات اعمال کرده و آن را بررسی کنیم. نتایج اعمال این تبدیل را به‌سادگی می‌توان در هیستوگرام‌های شکل ۵–۱ مشاهده نمود که اطلاعات تقریباً به‌صورت توزیع نرمال درآمده‌اند. حال به‌سادگی با کمک این قاعده که ۹۵.۴ درصد اطلاعات در توزیع نرمال در بازه‌ی میانگین و دو برابر انحراف معیار قرار می‌گیرند، اطلاعاتی که خارج از این محدوده هستند را حذف نمود.



شکل ۵–۹- نمودار تعداد و توزیع این رویدادها نسبت به یکدیگر بعد از انجام تبدیل box-cox

پس از حذف این اطلاعات و جایگزین کردن اطلاعات خارج از محدوده با بزرگ‌ترین نمونه داخل محدوده، شکل ۵–۹ به شکل ۵–۱۰ تبدیل می‌شود. با این روش داده‌های پرت را حذف نمودیم تا بتوانیم مقایسه‌ای عادلانه در میان رانندگان انجام دهیم. در بخش بعدی از این اطلاعات استفاده می‌کنیم تا بتوانیم مدلی برای نمره دهی به رفتار رانندگان پیاده‌سازی کنیم.

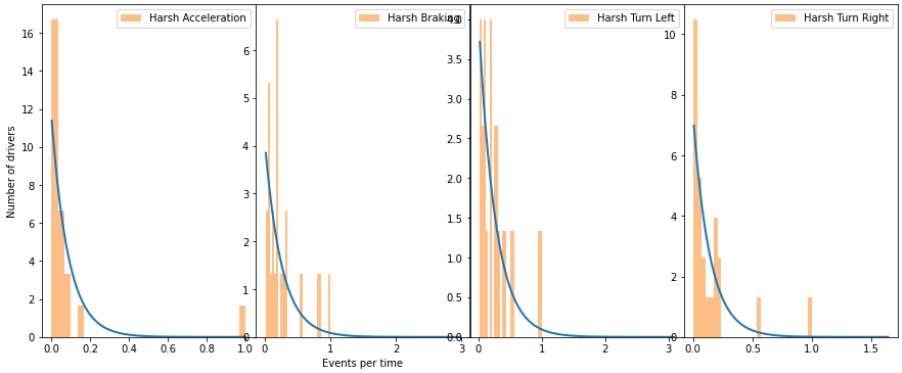


شکل ۵–۱۰- نمودار تعداد و توزیع این رویدادها نسبت به یکدیگر بعد از جایگزین کردن دادگان پرت

۵-۳ طراحی مدل

فرض اولیه ما هنگام تجزیه‌وتحلیل این داده‌ها این است که راننده‌ای که رخدادهای خطرناک بیشتری در زمان مشخص تولید کند، به‌احتمال بیشتری نسبت به راننده‌ای که رخدادهای خطرناک کمتری ایجاد کرده، خطرناک است. همچنین راننده‌ای که خطرناک‌تر است باید نمره کمتری بگیرد و راننده‌ای که خطرناک رانندگی نکرده و ایمن است باید امتیاز بالاتری بگیرد. به همین منظور سعی کردیم راننده را با سایر رانندگان مقایسه کنیم. برای این کار از اطلاعات تمیز شده بخش قبل استفاده کرده و به هر نوع رویداد (ترمز خطرناک، شتابگیری خطرناک، گردش‌به‌چپ خطرناک، گردش‌به‌راست خطرناک) یک توزیع نسبت دهیم و سپس هر راننده را با میانگین اطلاعات جامعه آماری مقایسه کنیم.

همان‌طور که در هیستوگرام شکل ۵–۱۰ مشاهده می‌شود، در آن‌ها توزیع به گوشه کشیده شده است و شکلی همچون توزیع گاما یا توزیع نمایی به خود گرفته است. پس هرکدام از رخدادهای فوق را می‌توان با یک توزیع نمایی مدل کرد و تابع چگالی احتمالی[۱] از جنس توزیع نمایی به هرکدام اختصاص داد.



شکل ۵–۱۱- نمودار اعمال توزیع نمایی به هرکدام از رویدادها

در جدول ۵-۱ میزان مجموع مربعات ناشی[۲] از این تخصیص قابل‌مشاهده است:

جدول ۵–۱	Harsh Acceleration	Harsh Braking	Harsh Turn Left	Harsh Turn Right
SSE	۴۸۸.۷۴۲۴	۱۷۸.۹۹۶۰	۱۰۵.۴۹۳۲	۱۲۸.۶۰۲۶

[۱] PDF

[۲] sum of squared estimate of errors (SSE)

سپس با کمک تابع چگالی احتمال فوق، تابع توزیع احتمال جمعی[1] هر رویداد را حساب می‌کنیم تا به بیان واضح‌تر میزان احتمال ایجاد هر رخداد به دست بیاید. در ادامه، قابل‌فهم‌ترین راهکار این است که برای پیش‌آمد هر رویداد (رفتار راننده‌ها)، میزان این احتمالات را با یکدیگر جمع کنیم تا پارامتری از میزان ایمن بودن افراد به دست آید. به‌طور مثال همان‌طور که در جدول ۵-۲ مشاهده می‌کنید، راننده ۶۳با امتیاز ۰.۵۴۱۷ ایمن‌ترین راننده است و دلیل آن، تنها وجود تعدادی رخداد ترمزگیری خطرناک است و با توجه به جدول ۵-۳ خطرناک‌ترین راننده، راننده ۴۹ با امتیاز ۳.۰۱۴۴ است.

جدول ۵-۲- لیست ایمن‌ترین رانندگان به‌صورت جمع احتمال

DriverId	Harsh Acceleration_CDF	Harsh Braking_CDF	Harsh Turn Left_CDF	Harsh Turn Right_CDF	metric
63	0.0000	0.5097	0.0000	0.0321	0.5417
14	0.0000	0.4493	0.2089	0.0630	0.7212
80	0.0000	0.3693	0.2399	0.1507	0.7599
75	0.3948	0.0000	0.2972	0.0934	0.7853
6	0.0000	0.0440	0.6650	0.2038	0.9128

جدول ۵-۳- لیست خطرناک‌ترین رانندگان به‌صورت جمع احتمال

DriverId	Harsh Acceleration_CDF	Harsh Braking_CDF	Harsh Turn Left_CDF	Harsh Turn Right_CDF	metric
51	1.0000	0.1867	0.0753	0.9991	2.2611
73	0.0000	0.7186	0.9682	0.9788	2.6657
8	0.5810	0.6100	0.8467	0.7616	2.7993
44	0.1464	0.9541	0.9746	0.7456	2.8207
49	0.5586	0.9627	0.7736	0.7196	3.0144

هرچند شاید این روش ساده به نظر بیاید ولی در عمل بسیار کارآمد است. درصورتی‌که پارامترmetric ، عدد صفر را برگرداند به این معنا است که راننده هیچ رویداد خطرناکی ایجاد نکرده است و راننده‌ای که به عدد چهار را برسد به این معنا است که بیشترین میزان ممکن خطا را ایجاد کرده است.

برای بهتر معنی دادن پارامتر metric که تعریف شد، این عدد را با یک تابع تبدیل، بین ۰ و ۱۰ آورده و آن را قرینه می‌کنیم تا رانندگانی که عملکرد خطرناک‌تری دارند، امتیاز کمتر بگیرند و مفهوم امتیاز رانندگان بهتر تعریف شود. بدین شکل امتیاز رانندگان محاسبه خواهد شد. در جدول ۵-۴ و جدول ۵-۵ می‌توان امتیاز خطرناک‌ترین و ایمن‌ترین رانندگان را مشاهده نمود.

[1] CDF

DriverId	Harsh Acceleration_CDF	Harsh Braking_CDF	Harsh Turn Left_CDF	Harsh Turn Right_CDF	metric	score
63	0.0000	0.5097	0.0000	0.0321	0.5417	8.6457
14	0.0000	0.4493	0.2089	0.0630	0.7212	8.1970
80	0.0000	0.3693	0.2399	0.1507	0.7599	8.1002
75	0.3948	0.0000	0.2972	0.0934	0.7853	8.0368
6	0.0000	0.0440	0.6650	0.2038	0.9128	7.7180

جدول ۵–۴- لیست ایمن‌ترین رانندگان به‌صورت امتیاز انتقال داده‌شده

DriverId	Harsh Acceleration_CDF	Harsh Braking_CDF	Harsh Turn Left_CDF	Harsh Turn Right_CDF	metric	score
51	1.0000	0.1867	0.0753	0.9991	2.2611	4.3472
73	0.0000	0.7186	0.9682	0.9788	2.6657	3.3357
8	0.5810	0.6100	0.8467	0.7616	2.7993	3.0018
44	0.1464	0.9541	0.9746	0.7456	2.8207	2.9482
49	0.5586	0.9627	0.7736	0.7196	3.0144	2.4639

جدول ۵–۵- لیست خطرناک‌ترین رانندگان به‌صورت امتیاز انتقال داده‌شده

در این شیوه تحلیل، وزن همه‌ی رویدادهای خطرناک یکسان در نظر گرفته‌شده و تفاوتی میان رویدادهای ایجادشده قائل نمی‌شود و این ممکن است زیاد برای مقایسه رانندگان منصفانه نباشد؛ اما با مجموعه دادگان فعلی، نمی‌توان اطلاعات ارزشمندتری کسب نمود، بهتر است مجموعه دادگان دیگری از اطلاعات تصادفات جمع‌آوری کرده تا وزن‌دهی میان رویدادهای خطرناک را با توجه به آن به دست آورد. بدین ترتیب هنگام محاسبه امتیاز رانندگان می‌توان برای مثال تأثیر شتابگیری خطرناک را بیشتر از ترمزگیری خطرناک محاسبه نمود ولی این کار نیازمند پژوهش و اطلاعات بیشتری است.

حال با توجه به توزیع به‌دست‌آمده برای بهره‌گیری آن در عمل و بررسی عملکرد رانندگان جدید باید ابتدا و انتهای حرکت هر راننده را که ذخیره‌شده به دست آوریم. سپس زمان رانندگی راننده را محاسبه نموده و تعداد رویدادها خطرناکی را که ایجاد نموده محاسبه کنیم تا بردار ویژگی هر راننده محاسبه شود. سپس با توجه به توزیعی که روی دادگان به دست آمد، میزان امتیاز راننده را حساب کنیم.

به‌طور مثال دو راننده‌ی ایمن و خطرناک در نظر بگیرید که ساعت‌ها رانندگی کرده‌اند و تعدادی رویداد خطرناک از آن‌ها در سرور جمع‌آوری‌شده و قصد داریم به آن‌ها امتیاز دهیم. ابتدا زمان رانندگی و تعداد خطاهای رانندگی آن‌ها را حساب می‌کنیم که برای راننده ایمن بردار ویژگی معادل:

['Harsh Acceleration':۰,۰۰۵,'Harsh Braking':۰,۰۰۵,'Harsh Turn Left':۰,۰۰۵,'Harsh Turn Right':۰,۰۰۵]

و برای راننده خطرناک بردار ویژگی معادل:

['Harsh Acceleration':۰,۵,'Harsh Braking':۰,۷,'Harsh Turn Left':۰,۳,'Harsh Turn Right':۰,۹]

به دست می‌آید، حال با کمک توزیع بکار گرفته‌شده، و شیوه‌ی محاسبه امتیاز، نتایج زیر به دست می‌آید:

جدول ۵–۶–امتیاز ۹.۹ برای راننده ایمن

	Harsh Acceleration	Harsh Braking	Harsh Turn Left	Harsh Turn Right	metric	score
0	0.0307	0.0000	0.0000	0.0026	0.0333	9.9168

جدول ۵–۷–امتیاز ۱ برای راننده خطرناک

	Harsh Acceleration	Harsh Braking	Harsh Turn Left	Harsh Turn Right	metric	score
0	0.9967	0.9297	0.6492	0.9982	3.5738	1.0655

٤-٥ جمع‌بندی

در این فصل با به¬کارگیری مجموعه دادگانی دیگر که اطلاعات رانندگان بیشتری را در اختیار ما قرار می‌داد، مکانیزمی برای نمره دهی به رانندگان بر اساس تعداد و تنوع رخداد در طول مدت رانندگی، بر اساس توزیع رفتارهای رانندگان پرداختیم و نشان دادیم که رانندگان جدید که اطلاعات خود را به سیستم دریافت و ذخیره‌سازی اطلاعات، ارسال می‌کنند را می‌توان نمره دهی و مقایسه نمود. حال در فصل بعد به انجام آزمایشات متعددی بر روی سیستم طراحی شده در سه فصل اخیر خواهیم پرداخت و عملکرد آن را بررسی خواهیم نمود.

٦ فصل ششم

آزمایشات و شبیه‌سازی

آزمایشات و شبیه‌سازی

پیش‌تر، به طراحی مدلی برای تشخیص رخدادهای ایجادشده توسط رانندگان و پیاده‌سازی محیطی برای دریافت و ذخیره‌سازی اطلاعات و انجام فرایند امتیازدهی به رانندگان پرداختیم. در این فصل بخش‌های فوق را موردبررسی و آزمایش دقیق‌تری قرار خواهیم داد. ابتدا مدل تشخیص رخدادهای خطرناک راننده که از دو حسگر شتاب سنج و ژیروسکوپ به دست آمد را بررسی کرده و اهمیت هرکدام از حسگرها را می‌سنجیم. همچنین ارزیابی می‌کنیم که کدام‌یک از حسگرها اطلاعات مفیدتری در اختیار ما قرار می‌دهد. سپس اهمیت شیوه پنجره‌بندی به‌صورت اتفاقی را بررسی کرده و نتایج آن را باحالتی که پنجره‌بندی به‌صورت اتفاقی رخ نداده مقایسه می‌کنیم. در پایان نیز فرایند دریافت و ذخیره‌سازی اطلاعات را با کمک طراحی یک برنامه که تولنایی ایجاد چندین ارسال‌کننده اطلاعات به سرور دارد، پیاده‌سازی کرده تا اتفاقی که هنگام عملی شدن پروژه رخ می‌دهد را شبیه‌سازی کنیم.

٦-١ بررسی اهمیت حسگرها

از بخش اول فصل سوم به یاد داریم، برای تشخیص رخدادهای خطرناک، می‌بایست سه مرحله را طی کنیم؛ ابتدا فرایند پنجره‌بندی اتفاقی را انجام دهیم، سپس با کمک الگوریتم DTW میزان شباهت این پنجره‌ها را از رویدادهای برچسب خورده به دست آوریم و در ادامه با کمک یک درخت تصمیم‌گیری، مدلی برای تشخیص رویداد به دست آورده و میزان دقت آن را محاسبه کنیم تا با توجه به پارامترهای بکار رفته در پنجره‌بندی اتفاقی، بهترین مدل انتخاب شود. در بخش قبل این فرایند بر روی اطلاعات حسگرهای شتابسنج و ژیروسکوپ بود. در این بخش قصد داریم فرایند آموزش را به‌صورت تک‌به‌تک برای هرکدام از حسگرها انجام دهیم و نتایج را با مدلی که در فصل قبل طراحی شد، مقایسه کنیم تا به اهمیت هرکدام از حسگرها برای تشخیص هر یک از این رویدادها بپردازیم.

در جدول ۶-۱ می‌توانیم میزان دقت مدل برای پنجره‌هایی با طول‌های ۲۲۶، ۱۲۶، ۱۸۶ با پارامتر احتمال نگاه به عقب و میانگین حرکت به جلوی متفاوت، تنها بر روی اطلاعات حسگر شتاب سنج مشاهده کنیم. میزان دقتی که از این جدول درج‌شده برابر میانگین دقت f۱-score برای اتفاق هرکدام از رخدادها است. با بررسی این جدول می‌توان دریافت بهترین مدل‌هایی که در این جدول قرارگرفته‌اند،

میانگین حرکت به جلوی ۰.۰۵ دارند به این معنا که پنجره‌ها با سـرعت کمی روی سـری زمانی حرکت می‌کنند و پارامتر احتمال نگاه به عقب در آن برای بهترین مدل‌ها برابر ۱۰ و ۲۰ درصد است.

جدول ۶–۱–میزان دقت به ازای مدل‌های آموزش داده‌شده، به ازای پنجره‌های متفاوت بر روی اطلاعات حسگر شتابسنج

F1-score Macro avg on ۲۲۶		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۸۰	۷۰	۶۹	۶۹
	٪۲۰	۷۷	۷۱	۷۵	۷۰
	٪۳۰	۷۶	۷۶	۷۶	۷۴
F1-score Macro avg on ۱۲۶		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۷۱	۶۸	۷۱	۷۲
	٪۲۰	۷۶	۶۸	۶۶	۷۰
	٪۳۰	۷۳	۷۲	۷۰	۷۰
F1-score Macro avg on ۱۸۶		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۸۸	۹۰	۸۵	۸۶
	٪۲۰	۹۲	۹۰	۸۶	۸۷
	٪۳۰	۹۰	۸۴	۸۷	۸۵

مقادیر شیوه یافتن بهترین مدل در مقایسه با جدول ۳-۱۰ (نتایج دقت ناشی از هر دو حسگر شتابسنج و ژیروسکوپ در فصل قبل) برای نوع میانگین حرکت متحرک یکسان و برابر ۰.۰۵ است ولی احتمال نگاه

به عقب برای بهترین مدل‌ها، کمی متفاوت است، بر اساس جدول ۳-۱۰ بهترین مدل برای پنجره‌هایی به طول ۲۲۶، عدد ۳۰ درصد بوده، درحالی‌که در اینجا برابر ۱۰ درصد شده است، به‌طور مشابه، برای پنجره‌هایی به طول ۱۲۶ و ۱۸۶ نیز به ترتیب از ۱۰ درصد و ۳۰ درصد به ۲۰ درصد تغییریافته به این معنا که برای یافتن بهترین مدل باید احتمال نگاه به عقب‌های متفاوت را به‌طور کامل بررسی نمود. چراکه می‌تواند به‌صورت مستقیم بر مدل انتخابی تأثیر بگذارد.

حال به سراغ مقایسه دقت‌ها می‌رویم و بهترین مدل از هرکدام را با یکدیگر مقایسه می‌کنیم؛

در پنجره‌هایی به طول ۲۲۶ که به دنبال رویدادهای شتابگیری خطرناک، چرخش به چپ و راست خطرناک هستیم، با توجه به جدول ۳-۱۰، دقت کلی ۸۴ درصد داریم که با نگاه دقیق‌تر در جدول۳-۱۱-الف، میزان دقت f۱-score برای رویداد چرخش به چپ خطرناک ۸۳ درصد و رویداد گردش‌به‌راست خطرناک ۹۱ درصد و رویداد شتابگیری خطرناک ۶۹ درصد بوده، این درحالی‌که است که همان‌طور که در جدول۶-۲-الف مشاهده می‌کنید، این اعداد به ترتیب به ۷۶، ۸۳ و ۶۱ درصد کاهش‌یافته است. دلیل این کاهش دقت را می‌توان به عدم استفاده از اطلاعات حسگر ژیروسکوپ نسبت داد.

در پنجره‌هایی به طول ۱۲۶ که به دنبال رویدادهای تغییرلاین خطرناک به چپ و راست هستیم، با توجه به جدول ۳-۱۰، دقت کلی ۸۱ درصد داریم که با نگاه دقیق‌تر در جدول۳-۱۱-ب، میزان دقت f۱-score برای رویداد تغییرلاین به چپ خطرناک ۷۳ درصد و رویداد تغییرلاین به راست خطرناک ۶۹ درصد بوده، این درحالی‌که است که همان‌طور که درجدول۶-۳-ب مشاهده می‌کنید، این اعداد به ترتیب به ۷۰ و ۵۷ درصد تغییریافته است. دلیل این کاهش دقت را می‌توان به عدم استفاده از اطلاعات حسگر ژیروسکوپ نسبت داد ولی همان‌طور که قابل‌بررسی است این کاهش دقت زیاد نیست، این را این‌گونه می‌توان تفسیر کرد که میزان عملکرد اطلاعات حسگر شتابسنج در مقابل با حسگر ژیروسکوپ در تشخیص رخدادهای تعویض لاین به راست و تعویض لاین به چپ بیشتر است.

در پنجره‌هایی به طول ۱۸۶ که به دنبال رویدادهای ترمزگیری خطرناک هستیم، با توجه به جدول ۳-۱۰، دقت کلی ۹۰ درصد داریم که با نگاه دقیق‌تر در جدول ۳-۱۱-ج، میزان دقت f۱-score برای رویداد ترمزگیری خطرناک ۸۱ درصد بوده، این درحالی‌که است که همان‌طور که در جدول ۶-۲-ج مشاهده می‌کنید، این به ۸۴ درصد افزایش‌یافته است. این افزایش را این‌گونه می‌توان تفسیر کرد که

میزان عملکرد اطلاعات حسگر ژیروسکوپ در تشخیص رخدادهای ترمزگیری خطرناک منجر به کاهش دقت می‌شود و از اطلاعات این سنسور نباید بهره گرفت.

جدول ۶-۲- اطلاعات دقیق میزان دقت نقاط برگزیده‌شده در جدول ۶-۱

الف

window is 226

	precision	recall	f1-score	support
NAG	0.97	0.99	0.98	3762
aceleracao_agressiva	0.87	0.47	0.61	131
curva_direita_agressiva	0.83	0.82	0.83	104
curva_esquerda_agressiva	0.86	0.69	0.76	108
accuracy			0.96	4105
macro avg	0.88	0.74	0.80	4105
weighted avg	0.96	0.96	0.96	4105

ب

window is 126

	precision	recall	f1-score	support
NAG	0.99	1.00	1.00	7289
troca_faixa_direita_agressiva	0.96	0.40	0.57	62
troca_faixa_esquerda_agressiva	0.90	0.57	0.70	47
accuracy			0.99	7398
macro avg	0.95	0.66	0.76	7398
weighted avg	0.99	0.99	0.99	7398

ج

window is 186

	precision	recall	f1-score	support
NAG	1.00	1.00	1.00	4880
freada_agressiva	0.88	0.80	0.84	116
accuracy			0.99	4996
macro avg	0.94	0.90	0.92	4996
weighted avg	0.99	0.99	0.99	4996

حال که مدل را تنها با اطلاعات حسگر شتاب سنج بررسی کردیم و به نتایج جالبی دست‌یافتیم وقت آن رسیده که همین کار را با اطلاعات حسگر ژیروسکوپ انجام دهیم. در جدول ۶-۳ می‌توانیم میزان دقت مدل برای پنجره‌هایی با طول‌های ۲۲۶، ۱۲۶، ۱۸۶ با پارامتر احتمال نگاه به عقب و میانگین حرکت به جلوی متفاوت بر روی اطلاعات حسگر شتاب سنج مشاهده کنیم. از مقایسه مقادیر نوع یافتن بهترین مدل این بخش در مقایسه با جدول ۳-۱۰، می‌توان دریافت که نوع انتخاب بهترین مدل با آن جدول دقیقا یکسان است که این‌گونه می‌توان تفسیر کرد که این اطلاعات در انتخاب بهترین مدل نسبت به اطلاعات ژیروسکوپ مؤثرتر هستند.

جدول ۶–۳- میزان دقت به ازای مدل‌های آموزش داده‌شده، به ازای پنجره‌های متفاوت بر روی اطلاعات حسگر ژیروسکوپ

F۱-score Macro avg on ۲۲۶		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۷۲	۷۲	۶۵	۶۴
	٪۲۰	۷۲	۶۷	۶۷	۶۶
	٪۳۰	۷۴	۷۶	۷۳	۷۳
F۱-score Macro avg on ۱۲۶		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۷۷	۶۶	۷۵	۶۸
	٪۲۰	۷۲	۶۷	۶۴	۷۰
	٪۳۰	۷۱	۷۰	۷۹	۶۸
F۱-score Macro avg on ۱۸۶		μ			
		۰.۰۵	۰.۱	۰.۱۵	۰.۲
$Prob_{neg}$	٪۱۰	۷۳	۶۵	۶۸	۷۳
	٪۲۰	۷۶	۶۹	۶۸	۶۲
	٪۳۰	۷۸	۷۷	۶۶	۶۸

حال به سراغ مقایسه دقت‌ها می‌رویم و بهترین مدل از هرکدام را با یکدیگر مقایسه می‌کنیم؛

در پنجره‌هایی به طول ۲۲۶ که به دنبال رویدادهای شـــتابگیری خطرناک، چرخش به چپ و راســـت خطرناک هســـتیم، با توجه به جدول ۳-۱۰، دقت کلی ۸۴ درصـــد داریم که با نگاه دقیق‌تر در جدول ۳-۱۱-الف، میزان دقت f^1 برای رویداد چرخش به چپ خطرناک ۸۳ درصـــد و رویداد گردش‌به‌راســـت

خطرناک ۹۱ درصد و رویداد شتابگیری خطرناک ۶۹ درصد بوده، این درحالی‌که است که همان‌طور که در جدول ۶-۴-الف مشاهده می‌کنید، این اعداد به ترتیب به ۸۰، ۷۰ و ۵۰ تغییریافته است. این‌که در این مدل نتوانیم اطلاعات مربوط به‌شتاب گیری را تشخیص دهیم و کاهش دقت این‌قدر محسوس باشد بسیار طبیعی است چراکه این حسگر توانایی محاسبه نیروهای افقی را ندارد و این‌که میزان دقت در رویدادهای گردشی متناسب است، به این معنا است، درصورتی‌که بتوان به‌خوبی این اطلاعات را با اطلاعات حسگر شتابسنج ترکیب نمود، می‌توان به نتایج بهتری دست‌یافت.

در پنجره‌هایی به طول ۱۲۶ که به دنبال رویدادهای تغییرلاین خطرناک به چپ و راست هستیم، با توجه به جدول ۳-۱۰، دقت کلی ۸۱ درصد داریم که با نگاه دقیق‌تر در جدول ۳-۱۱-ب، میزان دقت f۱-score برای رویداد تغییرلاین به چپ خطرناک ۷۳ درصد و رویداد تغییرلاین به راست خطرناک ۶۹ درصد بوده، این درحالی‌که است که همان‌طور که جدول ۶-۴-ب مشاهده می‌کنید، این اعداد به ترتیب به ۶۶ و ۶۷ درصد کاهش‌یافته است. دلیل این کاهش دقت را می‌توان به عدم استفاده از اطلاعات حسگر شتابسنج نسبت داد

در پنجره‌هایی به طول ۱۸۶ که به دنبال رویدادهای ترمزگیری خطرناک هستیم، با توجه به جدول ۳-۱۰، دقت کلی ۹۰ درصد داریم که با نگاه دقیق‌تر در جدول ۳-۱۱-ج، میزان دقت f۱-score برای رویداد ترمزگیری خطرناک ۸۱ درصد بوده، این درحالی‌که است که همان‌طور که در جدول ۶-۴-ج مشاهده می‌کنید، این به ۵۸ درصد کاهش‌یافته است. به‌طور مشابه یا رویداد شتابگیری خطرناک، رویداد ترمزگیری خطرناک نیز رویدادی است که نمی‌توان با این حسگر یافت، پس انتظار دقت بالایی نیز نداشته‌ایم.

با بررسی هرکدام از حسگرها می‌توان به مجموعه اطلاعات جالبی دست‌یافت ولی به لحاظ بررسی اهمیت می‌توان گفت که حسگر شتابسنج از اهمیت بیشتری برخوردار است، چراکه با کمک آن می‌توان رویدادها را با دقت معمولی به دست آورد که با کمک اضافه کردن اطلاعات حسگر ژیروسکوپ میزان دقت را افزایش داد. درحالی‌که با تنها اطلاعات سنسور ژیروسکوپ می‌توان دو رویداد چرخش به چپ و راست را با دقت قابل قبولی تشخیص داد. ولی بهترین پیاده‌سازی استفاده از هر دو حسگر است.

جدول ۶-۴- اطلاعات دقیق میزان دقت نقاط برگزیده‌شده در جدول ۳-۱۰

الف

----------window is 226----------

	precision	recall	f1-score	support
NAG	0.96	0.99	0.98	3736
aceleracao_agressiva	0.73	0.38	0.50	128
curva_direita_agressiva	0.81	0.61	0.70	90
curva_esquerda_agressiva	0.90	0.72	0.80	98
accuracy			0.96	4052
macro avg	0.85	0.68	0.74	4052
weighted avg	0.95	0.96	0.95	4052

ب

----------window is 126----------

	precision	recall	f1-score	support
NAG	0.99	1.00	1.00	7297
troca_faixa_direita_agressiva	0.97	0.51	0.67	63
troca_faixa_esquerda_agressiva	0.89	0.52	0.66	46
accuracy			0.99	7406
macro avg	0.95	0.68	0.77	7406
weighted avg	0.99	0.99	0.99	7406

ج

----------window is 186----------

	precision	recall	f1-score	support
NAG	0.99	1.00	0.99	4823
freada_agressiva	0.81	0.45	0.58	114
accuracy			0.98	4937
macro avg	0.90	0.72	0.78	4937
weighted avg	0.98	0.98	0.98	4937

۶-۲ نتایج استفاده از پنجره‌بندی اتفاقی

همان‌طور که در فصل پیش بیان شد، سعی شد برای مکانیزم پنجره‌بندی، از نحوه نگاه کردن انسان به سری‌های‌زمانی ایده گرفته شود. این کار با کمک ۲ پارامتر میانگین سرعت حرکت به جلو و احتمال نگاه به عقب انجام شد. با دقت در پارامتر میانگین سرعت حرکت به جلوپنجره، برای بهترین مدل‌های به‌دست‌آمده در بخش قبل (حسگرها به‌صورت تکی) و فصل قبل (مدل طراحی‌شده با هر دو حسگر) متوجه می‌شویم که بهترین پارامتر میانگین سرعت حرکت به جلو برابر ۰.۰۵ است، به این معنا که در صورتی طول پنجره ۱۰۰ باشد، به‌صورت میانگین ۵ نمونه به جلو حرکت می‌کنیم. حالا در این بخش به‌جای استفاده از متغیر تصادفی که به‌صورت اتفاقی سیگنال را بررسی کند و به عقب و جلو نگاه بی اندازد به‌صورت ثابت پنجره را حرکت می‌دهیم بدین شکل که مشابه قبل سه پنجره با طول‌های ثابت را

با همان پارامتر ۰.۰۵ روی سیگنال حرکت می‌دهیم و سعی می‌کنیم دقت مدل را مانند روش‌های پیشین محاسبه کنیم، در جدول ۶-۵ می‌توانیم نتایج دقت برای این مدل از پنجره‌بندی را مشاهده کنیم.

جدول ۶-۵- میزان دقت درصورتی‌که از پنجره‌بندی اتفاقی استفاده نکنیم

الف

window is 226

	precision	recall	f1-score	support
NAG	0.97	0.99	0.98	3762
aceleracao_agressiva	0.87	0.47	0.61	131
curva_direita_agressiva	0.83	0.82	0.83	104
curva_esquerda_agressiva	0.86	0.69	0.76	108
accuracy			0.96	4105
macro avg	0.88	0.74	0.80	4105
weighted avg	0.96	0.96	0.96	4105

ب

window is 126

	precision	recall	f1-score	support
NAG	0.99	1.00	0.99	7297
troca_faixa_direita_agressiva	0.86	0.48	0.61	63
troca_faixa_esquerda_agressiva	0.62	0.43	0.51	46
accuracy			0.99	7406
macro avg	0.82	0.64	0.71	7406
weighted avg	0.99	0.99	0.99	7406

ج

window is 186

	precision	recall	f1-score	support
NAG	0.99	1.00	0.99	4882
freada_agressiva	0.91	0.65	0.76	137
accuracy			0.99	5019
macro avg	0.95	0.82	0.88	5019
weighted avg	0.99	0.99	0.99	5019

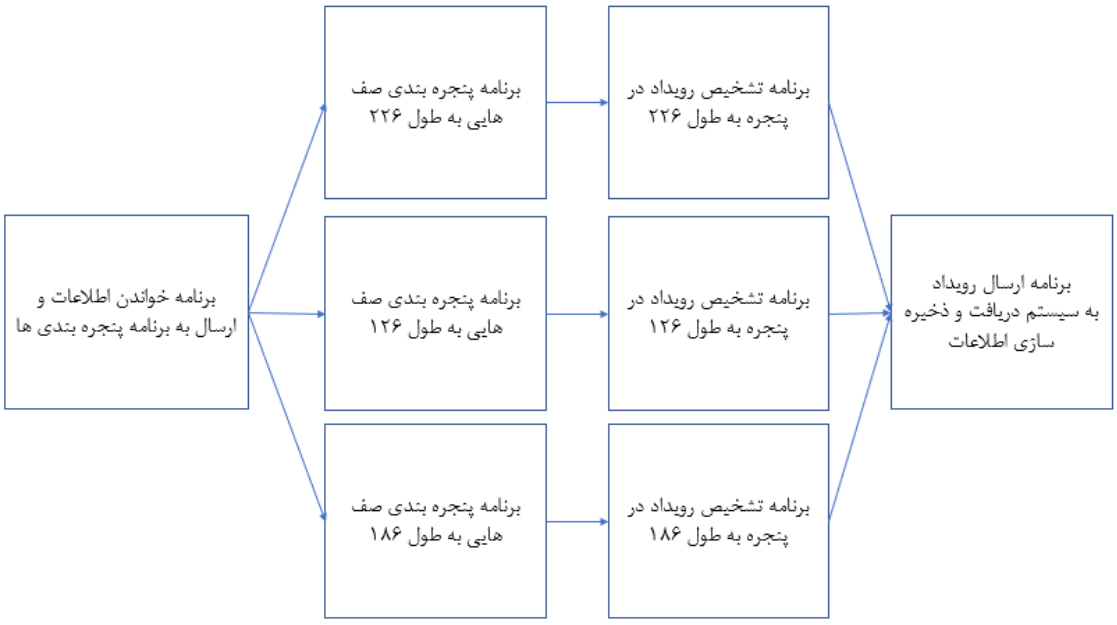
همان‌طور که قابل‌بررسی است درصورتی‌که از پنجره‌بندی اتفاقی استفاده نکنیم و سعی کنیم مدل را به‌صورت مستقیم بهره‌گیری کنیم، دقت به‌طور عمومی کاهش می‌یابد با نگاه دقیق‌تر و مقایسه جدول ۳-۱۱ و جدول۶-۵ دقت f۱-score برای داده‌های شتابگیری از ۶۹ درصد به ۶۱ درصد کاهش‌یافته برای دادگان گردش‌به‌راست ۸۷ درصد به ۸۳ درصد و گردش‌به‌چپ از ۸۳ درصد به ۷۶ درصد کاهش‌یافته است. به‌طور مشابه از مقایسه این دو جدول برای ردیف ب، می‌توان دریافت که میزان دقت برای رخداد تعویض لاین خطرناک به سمت راست از ۶۹ به ۶۱ درصد و تعویض لاین خطرناک به سمت چپ از ۷۳ درصد به ۵۱ درصد کاهش‌یافته است و در پایان برای پنجره‌هایی به طول ۱۸۶ که در ردیف ج جدول‌ها قرارگرفته‌اند، دقت ترمزگیری خطرناک از ۸۱ درصد به ۷۶ درصد کاهش‌یافته است. پس پرواضح است که

درصورتی‌که از این روش برای فرآیند آموزش استفاده نکنیم دقت کاهش محسوسی برای همه‌ی رویدادها حاصل می‌شود و انجام این حرکت تصادفی می‌تواند منجر به افزایش دقت مدل طراحی‌شده می‌شود.

۶-۳ شبیه‌سازی فرایند تشخیص و ارسال اطلاعات به ابرمحاسباتی

در فصل قبل به پیاده‌سازی سیستمی برای دریافت و ذخیره‌سازی اطلاعات بر بستر کانتینرهای داکر پرداختیم ولی آن را مورد تست قرار ندادیم، برای بررسی نحوه عملکرد سیستم دریافت و ذخیره‌سازی اطلاعات، یک شبیه‌ساز سیستم درون خودروها طراحی شد که درآن‌واحد بتواند به تعداد دلخواه راننده ایجاد کند و اطلاعات را بررسی کرده و به سرور ارسال کند. طراحی این بخش به‌نوعی شبیه‌سازی نوشتن برنامه‌ای است که باید بر روی سخت‌افزارها انجام شود. این شبیه‌سازی با زبان پایتون نوشته‌شده که به قابلیت به‌کارگیری و تست بر روی سخت‌افزارهایی همچون رزبری‌پای را دارد.

برای طراحی این نرم‌افزار از مکانیزم برنامه‌نویسی چند رشته‌ای استفاده شد. بدین شکل به ازای هر راننده برنامه‌هایی که در شکل ۶–۱ مشاهده می‌کنید اجرا خواهد شد. در ادامه جزئیات هر بخش از این برنامه بررسی خواهد شد.



شکل ۶–۱-برنامه‌ای که برای تست سیستم دریافت و ذخیره‌سازی اطلاعات به ازای هر راننده ساخته می‌شود.

ابتدا برنامه‌ای فعال می‌شود که اطلاعات را از حسگرها خوانده و به برنامه‌های پنجره‌بندی ارسال می‌کند.[1] در ادامه سه برنامه پنجره‌بندی برای طول پنجره‌های متفاوت ساخته‌شده که اطلاعات قرائت‌شده را متناسب با پارامتر λ که در فصل قبل تعریف و به دست آمد پیاده‌سازی می‌کند. این برنامه صبر می‌کند تا پنجره‌ها تکمیل شوند و هنگامی‌که پنجره‌ها تکمیل شد آن‌ها را به برنامه تشخیص رویداد مربوط به خود منتقل می‌کنند. پس از انتقال این رویداد، متناسب با پارامتر λ بهینه، تعدادی از نمونه‌ها حذف‌شده و مجدد صبر می‌شود تا نمونه‌ها درون پنجره پر شوند. در شکل ۶–۲ می‌توان رخداد فوق را برای پنجره به طول ۲۲۶ و پارامتر λ برابر ۰.۵ مشاهده کرد.

شکل ۶–۲–نحوه کنترل پنجره‌ها بر روی سری زمانی

در ادامه برنامه تشخیص رویداد متناسب با طول هر پنجره متناسب با درخت تصمیم‌گیری که در فصل قبل آموزش دادیم این برچسب مربوطه به پنجره تشخیص داده می‌شود و درصورتی‌که درون پنجره رویداد خطرناکی تشخیص داده شود، به برنامه انتقال به سیستم ذخیره‌سازی اطلاعات منتقل می‌شود.

بدین شکل می‌توانیم انتقال اطلاعات به سرور را شبیه‌سازی کنیم، در ادامه روند انتقال اطلاعات به سرور را بررسی می‌کنیم.

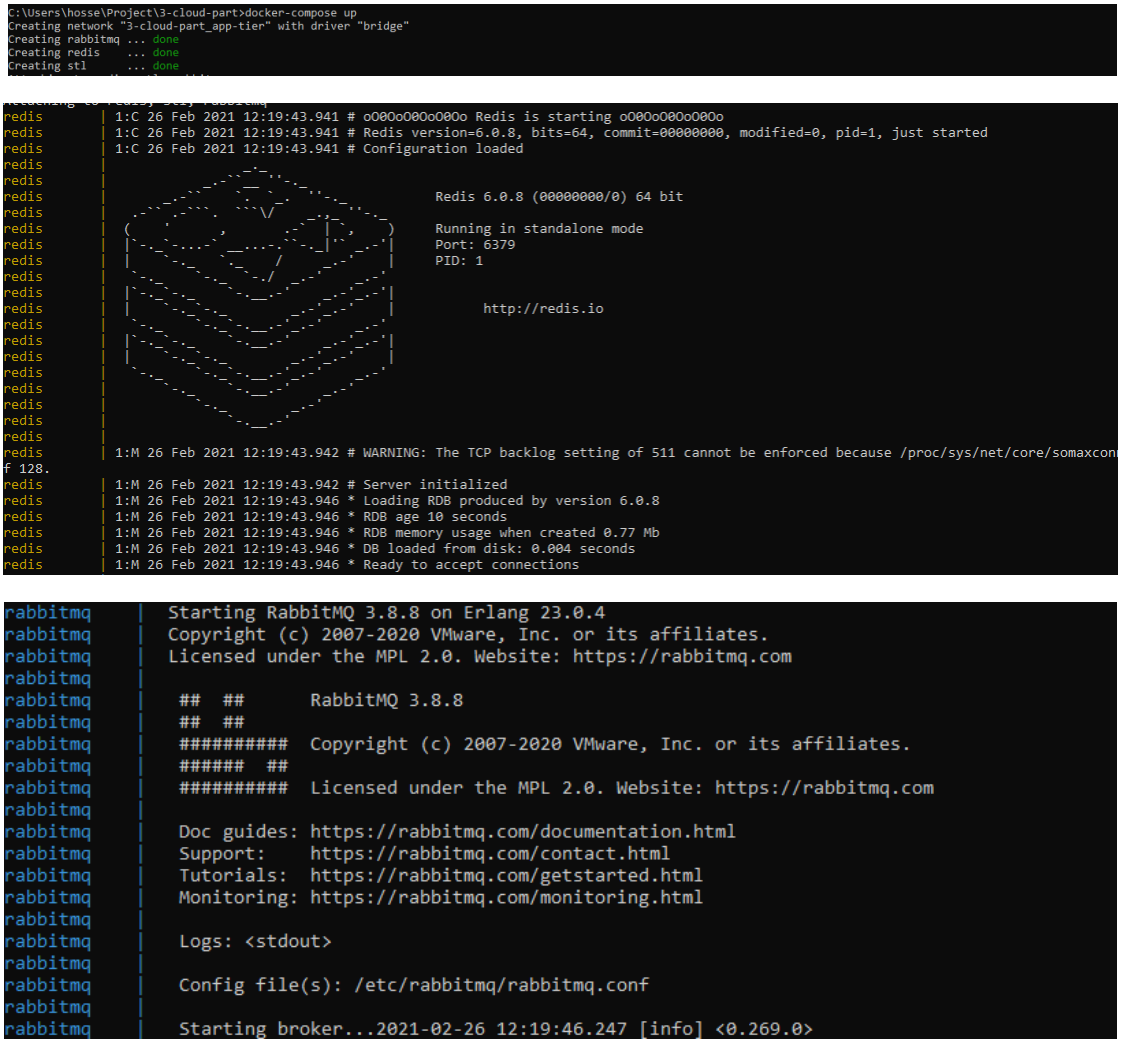
برای مدیریت بهتر اطلاعات در سرور و مشخص کردن زمان شروع زمان سفر و زمان پایان سفر در ابتدا و انتهای سفر بسته‌هایی با فرمت شکل به سرور ارسال می‌کنیم:

[1] این بخش به صورت شبیه سازی انجام شد و اطلاعات با فاصله زمانی های مشخصی از حافظه خوانده و به برنامه پنجره بندی منتقل شد.

```
'{"driver_id":'+str(id_)+',"timestamp":'+str(x)+' ,"event":"'+str('started')+'"}'
'{"driver_id":'+str(id_)+',"timestamp":'+str(x)+' ,"event":"'+str('Ended')+'"}
```

شکل ۶-۳- فرمت بسته‌های مدیریتی برای تشخیص ابتدا و انتهای سفر

برای نمایش روند انتقال اطلاعات به سیستم دریافت و ذخیره‌سازی اطلاعات، ابتدا می‌بایست آن را همان‌طور که در فصل پیش‌گفته شد، آماده کنیم، برای این کار تنها کافی است که دستور زیر را اعمال کنیم تا فرایند ساخته‌شدن بخش‌های مختلف به شکلی که در فایل docker-compose.yaml نوشته‌شده اجرا شود. برای این کار ابتدا کانتینر Redis اجراشده (زرد) سپس کانتینر RabbitMQ فعال‌شده (آبی) و در پایان برنامه‌ای که این دو را به هم متصل می‌کند اجراشده که به انتهای صف classic_queue_۱ در RabbitMQ متصل شده و اطلاعاتی که به این صف ختم شوند را به Redis منتقل می‌کند.



شکل ۶-۴- نحوه بالا آمدن سیستم دریافت و ذخیره‌سازی اطلاعات

پس از تکمیل و بالا آمدن سرویس‌دهنده، می‌توانیم برنامه مربوط به شبیه‌سازی تشخیص و پیاده‌سازی رویدادها را که پیش‌تر معرفی کردیم اجرا کنیم. خروجی این برنامه بسته‌هایی به سمت سرویس‌دهنده است که در شکل ۶–۵ می‌توانیم نمونه این بسته‌ها را برای راننده شماره ۱۷ مشاهده کنیم.

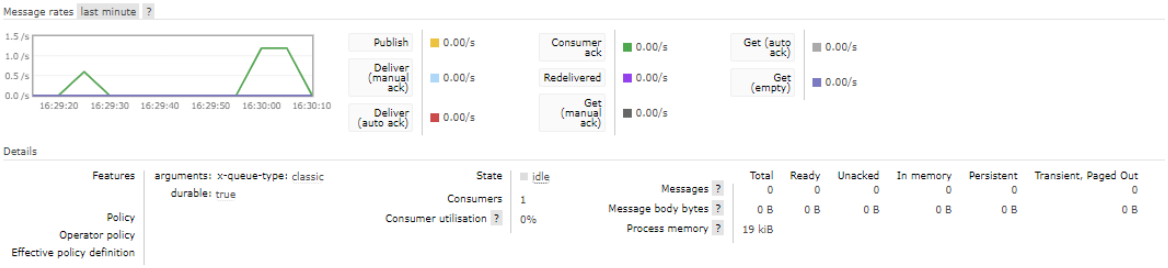
```
C:\Users\hosse\Project\4-Embeded-Simulator>python main.py
{"driver_id":17,"timestamp":1614342292 ,"event":"started"}
{"driver_id":17,"timestamp":1614342292 ,"event":"started"}
{"driver_id":17,"timestamp":1614342292 ,"event":"started"}
{"driver_id":17,"timestamp":"1970-01-01 00:00:14.378000","event":"Harsh_ChangeLine_Left"}
{"driver_id":17,"timestamp":"1970-01-01 00:00:15.006000","event":"Harsh_ChangeLine_Left"}
{"driver_id":17,"timestamp":"1970-01-01 00:00:23.801000","event":"Harsh_ChangeLine_Left"}
{"driver_id":17,"timestamp":"1970-01-01 00:00:27.571000","event":"Harsh_ChangeLine_Left"}
{"driver_id":17,"timestamp":"1970-01-01 00:02:24.497000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:02:25.400000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:02:34.431000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:02:35.334000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:02:47.976000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:02:48.879000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:02:49.782000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:42.157000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:43.060000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:43.963000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:44.866000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:46.476000","event":"Harsh_Turn_Right"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:49.774000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:56.605000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:57.508000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:58.411000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:03:59.668000","event":"Harsh_Turn_Right"}
{"driver_id":17,"timestamp":"1970-01-01 00:04:11.053000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:04:11.957000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:04:12.860000","event":"Harsh_Break"}
{"driver_id":17,"timestamp":"1970-01-01 00:04:16.157000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:05:07.826000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:05:08.926000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:05:10.024000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:03.893000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:04.992000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:06.091000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:22.581000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:23.680000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:24.779000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:25.879000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":"1970-01-01 00:06:40.169000","event":"Harsh_acceleration"}
{"driver_id":17,"timestamp":1614342866 ,"event":"Ended"}
9.607040977478027 min
```

شکل ۶–۵ بسته‌های ارسالی راننده ۱۷ به سیستم دریافت و ذخیره‌سازی اطلاعات

در ادامه در شکل ۶–۶ که مربوط به پنل مدیریت در RabbitMQ است، می‌توانیم نمودار دریافت اطلاعات مربوطه را برای راننده ۱۷ که نزدیک به ۱۰ دقیقه رانندگی کرده است را مشاهده کنیم.

در پایان نیز بااتصال مستقیم به پایگاه داده Redis با اجرای دستور redis-cli سعی می‌کنیم به آن دسترسی پیدا کنیم. سپس با اجرای دستور * KEYS می‌توانیم تمامی کلیدهای (رانندگانی) که در این پایگاه داده وجود دارند را مشاهده کنیم، سپس می‌توانیم با دستور HGETALL [۱۷] تمامی رخدادهایی

که توسط راننده ۱۷ تولیدشده را بررسی کنیم. در شکل ۶-۷، می‌توان رخدادهای ذخیره‌شده برای راننده شماره ۱۷ را در پایگاه داده Redis را مشاهده کرد.



شکل ۶-۶- پنل مدیریتی نرم‌افزار RabbitMQ حین دریافت اطلاعات از راننده شماره ۱۷

```
127.0.0.1:6379> HGETALL 17
 1) "1610893478"
 2) "started"
 3) "1970-01-01 00:00:14.378000"
 4) "Harsh_ChangeLine_Left"
 5) "1970-01-01 00:00:15.006000"
 6) "Harsh_ChangeLine_Left"
 7) "1970-01-01 00:00:23.801000"
 8) "Harsh_ChangeLine_Left"
 9) "1970-01-01 00:00:27.571000"
10) "Harsh_ChangeLine_Left"
11) "1970-01-01 00:02:24.497000"
12) "Harsh_Break"
13) "1970-01-01 00:02:25.400000"
14) "Harsh_Break"
15) "1970-01-01 00:02:34.431000"
16) "Harsh_Break"
17) "1970-01-01 00:02:35.334000"
18) "Harsh_Break"
19) "1970-01-01 00:02:47.976000"
20) "Harsh_Break"
21) "1970-01-01 00:02:48.879000"
22) "Harsh_Break"
23) "1970-01-01 00:02:49.782000"
24) "Harsh_Break"
25) "1970-01-01 00:03:42.157000"
26) "Harsh_Break"
27) "1970-01-01 00:03:43.060000"
28) "Harsh_Break"
29) "1970-01-01 00:03:43.963000"
```

شکل ۶-۷-نمونه رخدادهای ذخیره‌شده برای راننده شماره ۱۷ در پایگاه داده

٦-٤ مقایسه پژوهش فعلی با سایرین

از میان کارهای پیشین انجام شده که در فصل ۱ آمد، اطلاعات بکار رفته در دو پژوهش [٢٦] و [٢٧] که برای تشخیص رخدادهای خطرناک بر روی اطلاعات حسگر های شتابسنج و ژیروسکپ است، با مجموعه دادگان این پژوهش یکسان می‌باشد و توانایی مقایسه شدن با کار ما را دارد.

در [٢٦] که با کمک پنجره بندی ای ساده، تنها با محاسبه متغییر های آماری هر پنجره همچون واریانس، میانگین هر پنجره، و ضریبی از میانگین پنجره قبل سعی می‌کند با کمک طبقه بندی کننده های مختلف همچون شبکه عصبی، SVM و شبکه های بیزین رخدادهای خطرناک رانندگی را تشخیص دهد. که بیشترین دقت را یک با شبکه های عصبی با ۲ لایه مخفی و طول پنجره ای با ۲۰۰ نمونه تشخیص دهد. همچنین اهمیت سنسورهای بکار رفته در این پژوهش را بررسی کرده که بر اساس نتایج به این نکته دست یافته که میزان اهمیت حسگرهای شتاب‌سنج و ژیروسکوپ نسبت به مگنومتر بیشتر بوده و مدل بر اساس این دو حسگر رخدادها را تشخیص می‌دهند.

در [٢٧] شروع به پنجره بندی بر سیگنال های شتابسنج، ژیروسکپ و مگنومتر کرده و بجای اینکه بردار های ویژگی هر پنجره محاسبه شود، اطلاعات خام حسگر ها به شبکه های عصبی بازگشتی داده شده و سعی شده که با کمک شبکه های SimpleRNN،GRU و LSTM رخدادهای خطرناک رانندگی تشخیص داده شوند. که نتیجه کلی هر یک از شبکه ها به ترتیب ۹۴، ۷۰ ، ۹۵ درصد است. که شبکه عصبی LSTM با ۱۰ نرون بیشترین دقت را داشته است.

برتری این پژوهش نسبت به دو پژوهش فوق به شرح زیر است:

۱- ارائه یک شیوه جدید پنجره بندی که منجر به افزایش دقت شده و میتواند حتی در تکنیک های شبکه عصبی نیز مورد استفاده قرار گیرد و نتایج را بهبود ببخشد.

۲- طراحی مدلی متناسب با در نظر گرفتن زمان پردازش حین بکارگیری در عمل به نحوی که سیستم بتواند در زمان واقعی رخدادهای خطرناک رانندگی را تشخیص دهد.

۳- توانایی تشخیص رخدادهای رانندگی دیگری همچون حرکت مارپیچ و دور زدن کامل،.... که این درصورتی محقق خواهد شد که اطلاعات مربوطه را جمع آوری کنیم چرا که الگوریتم DTW میزان

شباهت الگو ها را بررسی می‌کند و در صورتی که برچسب رخدادهای فوق را داشته باشیم می‌توانیم آن‌ها را به سادگی با روش ارائه شده در این پژوهش بدست آوریم.

٤- همچنین در این پژوهش روند بررسی رفتار راننده از تشخیص رخداد تا تجمیع اطلاعات در سرور، و بررسی رفتار رانندگان انجام شد که دو گام نسبت به پژوهش های فوق فراتر رفته است.

همچنین می‌توان در مدل ارائه شده در این پژوهش، در بخش برچسب زدن پنجره‌ها پارامتری به مسئله اضافه کرد تا میزان اشتراک پنجره ها در نظر بگیرد، در حال حاضر درصورتی که پنجره رونده روی سیگنال با رخداد مربوطه اشتراک داشته باشند، پنجره مورد بررسی برچسب می‌خورد. این در حالی است، اگر تعریف کنیم، در صورتی که پنجره با رخداد واقعی حداقل ۵۰ درصد تداخل داشته باشند، و سپس برچسب خورده شود، دقت مدل حدود ۱۰ درصد افزایش می‌یابد و از دقت مدل شبکه عصبی نیز بهتر می‌شود. ولی بدلیل پیچیدگی در در بهینه‌سازی این پارامتر، این متغییر به مسئله اضافه نشد.

٦-٥ جمع‌بندی فصل چهارم

در این بخش به بررسی و آزمودن بخش‌های مختلف پروژه پرداختیم، ابتدا اهمیت هرکدام از حسگرها را با بررسی فرایند آموزش بر روی اطلاعات تنها یک حسگر انجام دادیم و دریافتیم که حسگر شتابسنج اهمیت بیشتری دارد. سپس با انجام آزمایشی دیگر بر روی اطلاعات هر دو حسگر، فرآیند آموزش را بدون به‌کارگیری پنجره‌بندی اتفاقی انجام دادیم و اثر آن را در تشخیص هر رخداد مشاهده نمودیم. در پایان نیز با کمک یک برنامه شبیه‌سازی، نحوه عملکرد سیستم دریافت و ذخیره‌سازی اطلاعات را در ابرمحاسباتی بررسی کردیم؛ و بسته‌هایی که یک راننده نمونه به این سیستم ارسال می‌کند را ارزیابی کردیم.

۷ فصل هفتم

نتیجه‌گیری و پیشنهادها

۷-۱ نتیجه‌گیری

در این رساله ما یک سیستم کامل برای پیاده‌سازی در شرکت‌های بیمه‌ای، استفاده در پلیس راهنمایی و رانندگی و شـرکت‌های حمل‌ونقل ارائه کردیم که با به‌کارگیری آن بتوان میزان ایمن بودن رانندگی افراد را سـنجید و در یک شـرکت بیمه‌ای، برای تعیین حق بیمه، در یک سـیسـتم عادلانه متناسـب با نوع رانـندگی افراد برای آن‌ها مبلغ، تعیین نمود تـا بتوان رانـندگـان را بـا انجـام این روش بـه رانـندگی ایمن تشـویق کرده تا جاده‌ها و خیابان‌هایی ایمن‌تری ایجاد شـود و آمار تصـادفات شـهری و تلفات جاده‌ای را کاهش یابد. برای این کار ابتدا مدلی با به‌کارگیری حسـگرهای شـتابسـنج و ژیروسـکوپ طراحی شـد که میزان رخدادهای خطرناک رانندگی با دقتی قابل‌قبول تشخیص دهند. رخدادهایی همچون گردش‌به‌راست خطرناک باسـقت ۹۱ درصـد (۸۷ درصـد[۱])، گردش‌به‌چپ خطرناک با دقت ۹۵ درصـد (۸۳ درصـد)، شـتابگیری خطرناک با دقت ۷۸ درصد (۶۹ درصد)، ترمزگیری خطرناک با دقت ۹۱ درصد (۷۳ درصد)، تعویض لاین خطرناک به سمت راست با دقت ۸۴ درصد (۶۹ درصد)، تعویض لاین خطرناک به سمت چپ با دقت ۸۳ درصد (۷۳ درصد) به‌دست‌آمده است. سپس یک سیستم دریافت و ذخیره‌سازی اطلاعات طراحی شد که بتواند بسته‌های تولیدی توسط خودروها را در یک سامانه ابری ذخیره‌سازی نماید. برای این کار از نرم‌افزار RabbitMQ به‌عنوان مبادله گر اطلاعات و Redis به‌عنوان پایگاه داده موقت استفاده شـد، همچنین برنامه با زبان پایتون نوشـته شـد که اطلاعات را از صـف‌های RabbitMQ خوانده و به Redis منتقل کند. کل این مجموعه بر بسـتر کانتینرهای داکر طراحی شـد تا تولنایی انتقال به هر ارائه‌دهنده سرویس ابری را داشته باشد. در پایان نیز یک مدل مبتنی بر اطلاعات آماری طراحی شد که به رانندگان بر اساس زمان رانندگی و تعداد رویدادهای خطرناک تولیدشده امتیازی اطلاق کند. تا بتوان رانندگان را با یکدیگر بر اسـاس این امتیاز مقایسـه کرد تا رانندگان ایمن را از رانندگان خطرناک متمایز کرده و حق بیمه‌ای متناسب با نوع رانندگی آن‌ها در نظر گرفت.

[۱] اعداد دقت اعلام شده دقت precision و اعداد داخل پرانتز معیار دقت f۱-score است.

۷-۲ پیشنهادها

در این بخش پژوهش‌هایی را که می‌توان در ادامه راه این پژوهش انجام شود ذکرشده:

۱-بهینه‌سازی با دقت بالاتر پارامترهای پنجره‌بندی اتفاقی برای تشخیص رخدادهای خطرناک

در این پروژه برای تشخیص رویدادهای خطرناک، از پنجره‌بندی اتفاقی استفاده شد که با کمک پارامترهای آن، چندین مدل آموزش داده‌شده‌اند و مدلی که دقت بیشتری بر روی دادگان تست در اختیار ما قرار می‌داد، به‌عنوان بهترین مدل انتخاب شد، ولی شیوه‌ای که برای تغییر این پارامترها بکار گرفته شد، برای پارامتر احتمال نگاه رو به عقب بافاصله‌ی ۱۰ درصد و برای پارامتر میانگین حرکت پنجره روبه‌جلو عدد ۵ درصد در نظر گرفته شد. به‌عنوان کارهای آینده می‌توان گستره بررسی را افزایش داد تا مدل‌های بیشتری موردبررسی قرار گیرند و نتایج بهتری به‌دست‌آید.

۲-استفاده از مجموعه دادگان دیگری برای امتیازدهی رانندگان بر اساس میزان اهمیت رخداد خطرناک

در فرایند نمره‌دهی به رانندگان، هنگامی‌که میزان احتمال خطرناک بودن رخدادها را محاسبه کردیم، این اعداد را به‌صورت مستقیم با یکدیگر جمع نمودیم و میزان خطرناک بودن، رخداد ترمزگیری خطرناک و گردش‌به‌چپ خطرناک را یکسان در نظر گرفتیم. درحالی‌که می‌توان با کمک مجموعه دادگانی دیگر در زمینه تصادفات میزان اهمیت هر یک از رخداد را به‌صورت دقیق اندازه گرفت و در فرایند نمره دهی به رانندگان بهره برد.

۳-استفاده از اطلاعات GPS و تشخیص رویدادهایی مربوط به‌سرعت غیرمجاز

برای فرایند تشخیص رخدادهای خطرناک، می‌توان با اضافه کردن حسگر GPS سرعت حرکت خودرو را محاسبه کرد و رخدادهای مربوط به‌سرعت غیرمجاز در محل را اندازه‌گیری نمود. همچنین با کمک این اطلاعات می‌توان محل‌های پرخطر را تشخیص داد.

۴-طراحی یک داشبورد برای نمایش اطلاعات و ارزیابی عملکرد رانندگان

در این پروژه داشبوردی برای نمایش اطلاعات پردازش‌شده، طراحی نشد، ولی یکی از بخش‌هایی که این پروژه نیاز دارد تا به مرحله عمل برسد، طراحی و پیاده‌سازی یک سرویس‌دهنده تحت وب است که اطلاعات پردازش‌شده و امتیازی که رانندگان کسب نموده‌اند را به آن‌ها اطلاع‌رسانی کند.

۸ مراجع

[۱] Wiegmann DA, Shappell SA, A Human Error Approach to Aviation Accident Analysis: The Human Factors Analysis and Classification System, London: Ashgate Publishing, ۲۰۰۳.

[۲] ف. احمدی, علل وقوع تصـــادفات در معابر شـــهری, تهران: پژوهشـــگاه علوم انســـانی و مطالعات فرهنگی, ۱۳۹۶.

[۳] Hickman, Jeffrey S., and E. Scott Geller, Self-management to increase safe driving among short-haul truck drivers, Journal of Organizational Behavior Management, ۲۰۰۵.

[٤] https://www.eghtesadnews.com/fa/tiny/news-۲۷۳۲۷٤, تعداد خودرو ها در ســـطح شهر, تهران: اقتصاد نیوز, ۱۳۹۷.

[٥] Ferreira, Jair, et al., Driver behavior profiling: An investigation with different smartphone sensors and machine learning., PLoS one ۱۲, no. ٤ : e۰۱۷٤۹٥۹., ۲۰۱۷.

[٦] Mohan, Prashanth, Venkata N. Padmanabhan, and Ramachandran Ramjee., Nericell: rich monitoring of road and traffic conditions using mobile smartphones., Proceedings of the ٦th ACM conference on Embedded network sensor systems., ۲۰۰۸.

[۷] Dai, Jiangpeng, et al., Mobile phone based drunk driving detection., ٤th International Conference on Pervasive Computing Technologies for Healthcare, pp. ۱-۸: IEEE, ۲۰۱۰.

[۸] White, Jules, et al., Wreckwatch: Automatic traffic accident detection and notification with smartphones., Mobile Networks and Applications ۱٦, no. ۳ : ۲۸٥-۳۰۳., ۲۰۱۱.

[٩] Araújo, Rui, et al., Driving coach: A smartphone application to evaluate driving efficient patterns., IEEE Intelligent Vehicles Symposium, pp. ١٠٠٥-١٠١٠: IEEE, ٢٠١٢.

[١٠] Eren H, Makinist S, Akin E, Yilmaz A, Estimating driving behavior by a smartphone, Intelligent Vehicles Symposium (IV), ٢٠١٢.

[١١] Fazeen, Mohamed, et al., Safe Driving Using Mobile Phones, IEEE Transactions on Intelligent Transportation Systems ١٣, no. ٣ : ١٤٦٢-١٤٦٨, ٢٠١٣.

[١٢] Castignani, German, et al., Driver behavior profiling using smartphones: A low-cost platform for driver monitoring., IEEE Intelligent Transportation Systems Magazine ٧,١ (٢٠١٥): ٩١-١٠٢.: IEEE, ٢٠١٥.

[١٣] Carvalho, Eduardo, et al., Exploiting the use of recurrent neural networks for driver behavior profiling., International Joint Conference on Neural Networks (IJCNN).: IEEE, ٢٠١٧.

[١٤] Eftekhari, Hamid Reza, Smartphone-based system for driver anger scale estimation using neural network on continuous wavelet transformation, AUT Journal of Mathematics and Computing ١, no. ١ : ١١٣-١٢٤., ٢٠٢٠.

[١٥] Cura, Aslihan, et al., Driver profiling using long short term memory (LSTM) and convolutional neural network (CNN) methods, IEEE Transactions on Intelligent Transportation Systems, ٢٠٢٠.

[١٦] Alamri, Atif, et al., An effective bio-signal-based driver behavior monitoring system using a generalized deep learning approach., IEEE Access ٨ (٢٠٢٠): ١٣٥٠٣٧-١٣٥٠٤٩, ٢٠٢٠.

[١٧] Chan, Stan Salvador and Philip, FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space, Florida Institute of Technology, ٢٠٠٧.

[١٨] Kruskall, J. & M. Liberman, The Symmetric Time Warping Problem: From Continuous to Discrete. In Time Warps,String Edits and Macromolecules: The Theory and Practiceof Sequence Comparison, ١٩٨٣: Addison-Wesley Publishing Co.

[١٩] J. Ferreira, Jr., et al, ‘Driver behavior profiling: An investigation with different smartphone sensors and machine learning, PLoS ONE, vol. ١٢, no.٤, ٢٠١٧.

[٢٠] Tanida, Kazuaki, Python implementation of FastDTW, https://pypi.org/project/fastdtw/, ٠,٣,٤ (٢٠١٩).

[٢١] svaante, decision-tree-id٣, https://pypi.org/project/decision-tree-id٣/, ٢٠٢١.

[٢٢] Docker overview, https://docs.docker.com/, ٢٠٢١.

[٢٣] https://www.rabbitmq.com/documentation.html, RabbitMQ documentation ٣,٨,٤, ٢٠٢١.

[٢٤] R.Rossi, Unsupervised driver safety estimation at scale, a collaboration with Pointer Telocation, https://devblogs.microsoft.com/cse/٢٠١٨/٠٧/٣٠/unsupervised-driver-safety-estimation-at-scale/, ٢٠١٨.

[٢٥] G.Box, D.Cox, An Analysis of Transformations, Journal of the Royal Statistical Society: Series B (Methodological), ١٩٦٤.

[٢٦] Ferreira, Jair, Eduardo Carvalho, Bruno V. Ferreira, Cleidson de Souza, Yoshihiko Suhara, Alex Pentland, and Gustavo Pessin, Driver behavior profiling: An investigation with different smartphone sensors and machine learning, PLoS one ١٢, no. ٤ : e٠١٧٤٩٥٩., ٢٠١٧.

[٢٧] Carvalho, Eduardo, Bruno V. Ferreira, Jair Ferreira, Cleidson De Souza, Hanna V. Carvalho, Yoshihiko Suhara, Alex Sandy Pentland, and Gustavo Pessin, Exploiting the Use of Recurrent Neural Networks for Driver Behavior Profiling, International Joint Conference on Neural Networks (IJCNN).: IEEE, ٢٠١٧.

[٢٨] Johnson, Derick A., and Mohan M. Trivedi., Driving style recognition using a smartphone as a sensor platform., ١٤th International IEEE Conference on Intelligent Transportation Systems (ITSC)., ٢٠١١.

[٢٩] Wahlström, Johan, Isaac Skog, and Peter Händel., Detection of dangerous cornering in GNSS-data-driven insurance telematics., IEEE Transactions on Intelligent Transportation Systems ١٦,٦ : ٣٠٧٣-٣٠٨٣., ٢٠١٥.

Abstract

Dangerous driving is one of the most important causes of car accidents. If drivers know that their behavior is being recorded while driving, they will be driving safer. Also, if it is possible to implement a system that encourages people to drive safely by being aware of the type of driving behaviors of individuals, this will lead to a reduction in traffic accidents. This project aims to implement a method for detecting driving events (including dangerous right and left lanes, dangerous lane changes left and right, dangerous braking, and acceleration) with the help of sensor information so that driver information Receive and store in a supercomputer and use a solution to calculate how safe people are driving. More precisely, using a new windowing method on time series, a decision tree on window similarities (using the Fast-DTW algorithm) was trained to detect driving events, then to receive and Data storage RabbitMQ was used as a message exchanger, and Redis was used as a temporary database. Finally, a statistical method was used to score and compare people's driving. The dangerousness of driving introduces them as a benchmark for comparison. With the help of the results presented in this study, an insurance plan can be implemented so that drivers who show safer driving pay fewer insurance premiums so that this plan can be used as an incentive for Improved driving culture was considered.

Key Words: Internet of Things, Machine learning, Driver Profiling, Driving culture

Amirkabir University of Technology
(Tehran Polytechnic)
Department of Electrical Engineering
MSc Thesis

Analyze information received from vehicles to investigate driver behavior in the computing cloud

By
Hossein Gholami
Supervisor
Dr. Seyed Ahmad Motamedi

Spring 2021