دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر
گروه سیستم‌های نرم‌افزاری

سنتز خط محصول‌های نرم‌افزاری برای ارزیابی تطبیقی یادگیری مدل

پروژه کارشناسی مهندسی کامپیوتر گرایش سیستم‌های نرم‌افزاری

فاطمه سیددباغی

استاد راهنما

دکتر رامتین خسروی

تیر ۱۴۰۲


بسم الله الرحمن الرحيم

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده‌ی برق و کامپیوتر
گروه سیستم‌های نرم‌افزاری

سنتز خط محصول‌های نرم‌افزاری برای ارزیابی تطبیقی یادگیری مدل

پروژه‌ی کارشناسی مهندسی کامپیوتر گرایش سیستم‌های نرم‌افزاری

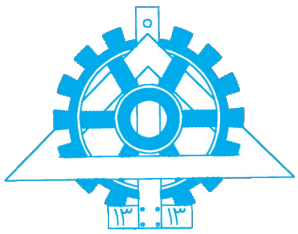
فاطمه سیددباغی

استاد راهنما

دکتر رامتین خسروی

تیر ۱۴۰۲

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



گواهی دفاع از پروژه کارشناسی

هیأت داوران پروژهٔ کارشناسی آقای / خانم فاطمه سیددباغی به شمارهٔ دانشجویی ۸۱۰۱۹۷۵۲۹ در رشتهٔ مهندسی کامپیوتر – گرایش سیستم‌های نرم‌افزاری را در تاریخ ..... با عنوان «سنتز خط محصول‌های نرم‌افزاری برای ارزیابی تطبیقی یادگیری مدل»

	به عدد	به حروف
با نمرهٔ نهایی		

و درجهٔ [ ] ارزیابی کرد.

ردیف	مشخصات هیأت داوران	نام و نام خانوادگی	مرتبهٔ دانشگاهی	دانشگاه یا مؤسسه	امضا
۱	استاد راهنما	دکتر رامتین خسروی	استادیار	دانشگاه تهران	
۲	استاد داور داخلی	دکتر فاطمه قاسمی اصفهانی	استادیار	دانشگاه تهران	

نام و نام خانوادگی معاون تحصیلات تکمیلی و پژوهشی دانشکده / گروه:

تاریخ و امضا:

نام و نام خانوادگی معاون آموزشی و تحصیلات تکمیلی پردیس دانشکده‌های فنی:

تاریخ و امضا:

تعهدنامهٔ اصالت اثر

باسمه تعالی

اینجانب فاطمه سیددباغی تأیید می‌کنم که مطالب مندرج در این پروژه حاصل کار پژوهشی اینجانب است و به دستاوردهای پژوهشی دیگران که در این نوشته از آن‌ها استفاده شده است مطابق مقررات ارجاع گردیده است. این پروژه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتری ارائه نشده است.

نام و نام خانوادگی دانشجو: فاطمه سیددباغی

تاریخ و امضای دانشجو:

قدردانی

در آغاز وظیفه خود می‌دانم از زحمات بی‌دریغ استاد راهنمای خود، جناب آقای دکتر رامتین خسروی، تشکر و قدردانی کنم که در طول انجام این پایان‌نامه با نهایت صبوری همواره راهنما من بودند و قطعاً بدون راهنمایی‌های ارزنده ایشان، این مجموعه به انجام نمی‌رسید.

از سرکار خانم شقایق توسلی که زحمت مشاوره، بازبینی و تصحیح این پایان‌نامه را تقبل فرمودند کمال امتنان را دارم.

از سرکار خانم دکتر فاطمه قاسمی که زحمت داوری این پایان‌نامه را تقبل فرمودند نیز کمال تشکر را دارم.

فاطمه سیددباغی

تیر ۱۴۰۲

چکیده

خط تولید نرم‌افزار مجموعه‌ای از سیستم‌های فشرده نرم‌افزار است که مجموعه‌ای مشترک و مدیریت شده از ویژگی‌ها، عملکردها و سایر ویژگی‌ها را به اشتراک می‌گذارد که نیازهای خاص یک بخش خاص از بازار را برآورده می‌کند و از مجموعه مشترکی از دارایی‌های اصلی توسعه می‌یابد. خط تولید نرم‌افزار اغلب در شرایطی استفاده می‌شود که نیاز به تولید خانواده‌ای از محصولات نرم‌افزاری مرتبط وجود دارد که از نظر ویژگی‌ها، عملکردها یا ویژگی‌های دیگر متفاوت هستند، اما یک هسته مشترک دارند. سیستم‌های نرم‌افزاری در طول چرخه‌ی عمر خود تغییرات متعددی را تجربه می‌کنند و به همین دلیل، مدل‌های آن‌ها ممکن است منسوخ شوند.

تنوع در یک سیستم نرم‌افزاری، تولید انبوه و سفارشی‌سازی را امکان پذیر می کند. برای به حداکثر رساندن این فرصت‌ها، باید سیستم‌های متغیر فشرده را با در نظر گرفتن متغیرها و اشتراکات به عنوان ویژگی‌های مهم این گونه سیستم‌های نرم‌افزاری به صورت ساختاریافته تحلیل کرد. مدل های رفتاری یکی از روش‌های کاربردی برای تجزیه و تحلیل رفتاری خطوط تولید نرم افزار، از جمله آزمایش و بررسی مدل‌ها هستند.

از آنجایی که روش‌های استفاده شده بر روی تعداد کمی از خطوط تولید نرم‌افزار آزمایش شده است، نتایج ممکن است با توجه به ویژگی‌های سیستم‌های موضوعی مغرضانه باشد. برای کاهش این تهدید، در این مطالعه، روشی برای تولید تصادفی خطوط تولید نرم افزار و بررسی ویژگی هایی که آنها را پیچیده تر می کند، پیشنهاد می کنیم. هدف ما شناسایی چالش‌برانگیزترین خطوط تولید نرم‌افزار است که می‌توانند برای ارزیابی و بهبود الگوریتم‌های یادگیری مدل در مطالعات آینده مورد استفاده قرار گیرند. برای دستیابی به این هدف، ما چارچوبی برای تولید آن‌ها با درجات مختلف پیچیدگی ایجاد می کنیم و از تجزیه و تحلیل آماری برای شناسایی ویژگی های کلیدی که بر پیچیدگی آنها تأثیر می گذارد، استفاده می کنیم. ما معتقدیم که یافته‌های ما می‌تواند به محققان و متخصصان کمک کند تا عواملی را که به پیچیدگی هاSPL کمک می‌کنند، درک کنند و تکنیک‌های مؤثرتری برای مدل‌سازی، یادگیری و مدیریت خطوط تولید نرم‌افزار توسعه دهند.

واژگان کلیدی مدل‌های رفتاری، خطور تولید نرم‌افزار، یادگیری مدل فعال، آزمون مبتنی برمدل

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

۱.۱ مقدمه

در سال‌های اخیر، پیچیدگی روزافزون خطوط تولید نرم‌افزار [۱] ، چالش‌های جدیدی را پیش روی مهندسان و محققان حوزه مهندسی نرم‌افزار قرار داده است. ماشین‌های حالت محدود [۲] (FSM) جزء اساسی خطوط تولید نرم‌افزار هستند و تأیید آن‌ها گامی اساسی در فرآیند توسعه است.

در حالی که تکنیک های تایید سنتی برای FSM ها، مانند شبیه سازی [۳] و تایید رسمی [۴] ، در بسیاری از موارد موفق بوده اند، آنها می توانند زمان بر و منابع فشرده باشند، به خصوص برای طراحی های پیچیده .FSM برای مقابله با این چالش ها، محققان و مهندسان به تکنیک های یادگیری ماشین برای خودکارسازی و بهینه سازی فرآیند تأیید FSM روی آورده اند.

در مطالعات قبلی، محققان از تکنیک‌های یادگیری ماشین برای پیش‌بینی تلاش تأییدیه [۵] مورد نیاز برای هاFSM در زمینه خطوط تولید نرم‌افزار استفاده کرده‌اند. با این حال، این مطالعات تنها به دو خط تولید نرم‌افزار محدود می‌شد که ممکن است نماینده همه خطوط تولید نرم‌افزاری ممکن یا سناریوهای دنیای واقعی نباشد.

برای پرداختن به این محدودیت، هدف مطالعه ما طراحی خطوط محصول نرم افزاری جدید برای تأیید

[1] Software product lines
[2] Finite state machines
[3] Simulation
[4] Formal verification
[5] Verification effort

الگوریتم های یادگیری مدل ما است که در فصل های بعدی توضیح داده شده است. برای اعتبار سنجی بهتر الگوریتم های خود، با ایجاد FSM های تصادفی و استخراج ویژگی ها از آنها شروع کردیم. با تجزیه و تحلیل این که کدام ویژگی ها به ما کمک کردند FSM های پیچیده تری تولید کنیم، می توانیم ویژگی های طراحی حیاتی را شناسایی کنیم که بر تلاش تأیید مورد نیاز برای FSM ها در خطوط تولید نرم افزار تأثیر می گذارد.

در این مقاله، ما یک رویکرد مبتنی بر یادگیری ماشین برای پیش‌بینی تلاش تأیید مورد نیاز برای هاFSM در خطوط تولید نرم‌افزار ارائه می‌کنیم. ما در مورد اهمیت تأیید FSM در خطوط تولید نرم‌افزار، چالش‌های مرتبط با آن و مزایای بالقوه استفاده از تکنیک‌های یادگیری ماشین برای بهبود کارایی و اثربخشی تأیید FSM در این حوزه بحث می‌کنیم.

یافته‌های ما پیامدهای مهمی برای طراحی و توسعه خطوط تولید نرم‌افزار پیچیده دارد و رویکرد ما می‌تواند به مهندسان نرم‌افزار در بهینه‌سازی فرآیند تأیید، کاهش زمان و هزینه مورد نیاز برای تأیید FSM و در نهایت افزایش کیفیت و قابلیت اطمینان خطوط تولید نرم‌افزار کمک کند.

۲.۱ مساله تحقیق

ماشین‌های حالت متناهی (FSM) به طور گسترده در خطوط تولید نرم‌افزار برای مدل‌سازی رفتار سیستم‌هایی با تعداد محدود حالت‌ها و ورودی‌ها استفاده می‌شوند.

با این حال، این مطالعات عمدتاً بر مجموعه محدودی از خطوط تولید نرم‌افزار و طرح‌های FSM تکیه کرده‌اند، که ممکن است نماینده همه خطوط تولید نرم‌افزار ممکن و سناریوهای دنیای واقعی نباشد. برای پرداختن به این محدودیت، هدف مطالعه ما ایجاد طرح‌های FSM تصادفی و استخراج ویژگی‌ها از آنها برای شناسایی ویژگی‌های طراحی حیاتی است که بر تلاش تأیید مورد نیاز برای هاFSM در خطوط تولید نرم‌افزار تأثیر می‌گذارد.

بنابراین، مشکل تحقیقی که در این مطالعه به آن پرداخته می‌شود، نحوه استفاده از تکنیک‌های یادگیری ماشین و انتخاب ویژگی برای پیش‌بینی تلاش تأیید مورد نیاز برای هاFSM در خطوط تولید نرم‌افزار است. با پرداختن به این مشکل، هدف ما بهبود کارایی و اثربخشی تأیید FSM در زمینه خطوط تولید نرم افزار، کاهش زمان و هزینه مورد نیاز برای تأیید FSM و در نهایت افزایش کیفیت و قابلیت اطمینان محصولات نرم افزاری است.

۳.۱ تاریخچه‌ای از موضوع تحقیق

در سال‌های اخیر، محققان و مهندسان بر خودکارسازی طراحی و آزمایش خطوط تولید نرم‌افزار برای بهبود کارایی و اثربخشی آنها تمرکز کرده‌اند. یکی از جنبه‌های مهم تست خط تولید نرم‌افزار، تأیید ماشین‌های حالت محدود (FSM) است که به طور گسترده برای مدل‌سازی رفتار سیستم‌های نرم‌افزاری استفاده می‌شوند.

برای پرداختن به چالش‌های راستی‌آزمایی FSM در زمینه خطوط تولید نرم‌افزار، محققان تکنیک‌های خودکار مختلفی مانند یادگیری ماشین، الگوریتم‌های ژنتیک و استخراج ویژگی‌ها را بررسی کرده‌اند. به عنوان مثال، مطالعه‌ای که در سال ۲۰۱۱ منتشر شد [۷] ، یک رویکرد مبتنی بر الگوریتم ژنتیک برای تولید هایFSM تصادفی برای آزمایش خط تولید نرم‌افزار پیشنهاد کرد. این رویکرد شامل استفاده از عملگرهای جهش [۶] و متقاطع [۷] برای تکامل هاFSM در طول نسل‌های مختلف و در عین حال ارزیابی تناسب آنها بر اساس یک معیار تعریف شده است. نتایج مطالعه نشان داد که رویکرد پیشنهادی، کارایی و اثربخشی تست خط تولید نرم‌افزار را در مقایسه با آزمایش دستی بهبود می‌بخشد.

به طور کلی، این مطالعات مزایای بالقوه استفاده از تکنیک‌های خودکار، مانند الگوریتم‌های ژنتیک و استخراج ویژگی، برای تولید و بهینه‌سازی هایFSM تصادفی برای تست خط تولید نرم‌افزار را نشان می‌دهند. این تکنیک‌ها می‌توانند به شناسایی ویژگی‌های طراحی حیاتی کمک کنند که بر تلاش تأیید مورد نیاز برای هاFSM در خطوط تولید نرم‌افزار تأثیر می‌گذارد، زمان و هزینه مورد نیاز برای تأیید را کاهش می‌دهد و در نهایت کیفیت و قابلیت اطمینان محصولات نرم‌افزاری را افزایش می‌دهد.

۴.۱ اهداف و آرمان کلی تحقیق

هدف اصلی تحقیق ایجاد چارچوبی برای تولید ماشین‌های حالت محدود تصادفی (FSM) و استخراج ویژگی‌ها از آنها برای شناسایی ویژگی‌های طراحی حیاتی است که بر تلاش مورد نیاز برای هاFSM در خطوط تولید نرم‌افزار تأثیر می‌گذارد. هدف این چارچوب بهبود کارایی و اثربخشی تست خط تولید نرم افزار با کاهش زمان و هزینه مورد نیاز برای تأیید FSM است.

[6] Mutation
[7] Crossover

برای دستیابی به این هدف، تحقیق ما به دنبال پاسخ به سؤالات تحقیق زیر است:

- چگونه می توانیم FSM های تصادفی را تولید کنیم که نماینده خطوط تولید نرم افزارهای دنیای واقعی هستند؟
- چگونه می‌توانیم ویژگی‌هایی را از هایFSM تصادفی استخراج کنیم تا ویژگی‌های طراحی حیاتی را که بر تلاش تأیید مورد نیاز برای هاFSM در خطوط تولید نرم‌افزار تأثیر می‌گذارند، شناسایی کنیم؟
- چگونه می توانیم از ویژگی های طراحی حیاتی شناسایی شده برای بهینه سازی فرآیند طراحی و آزمایش خطوط تولید نرم افزار و کاهش زمان و هزینه مورد نیاز برای تأیید FSM استفاده کنیم؟

هدف تحقیقاتی ما شامل کمک به دانش رو به رشد مربوط به استفاده از تکنیک های خودکار برای تست خط تولید نرم افزار است. به طور خاص، هدف تحقیق ما ارتقای پیشرفته‌ترین فناوری در تولید هایFSM تصادفی و تکنیک‌های استخراج ویژگی برای آزمایش خط تولید نرم‌افزار است. علاوه بر این، ارائه بینش ها و توصیه های عملی برای مهندسان خط تولید نرم افزار و محققانی است که علاقه مند به استفاده از تکنیک های خودکار برای بهینه سازی طراحی و آزمایش خطوط تولید نرم افزار هستند.

۵.۱ روش انجام تحقیق

این تحقیق بر توسعه الگوریتم‌های یادگیری ماشین موثر برای خطوط تولید نرم‌افزار متمرکز است. این شامل چندین مرحله کلیدی، از جمله تولید FSM های تصادفی و استخراج ویژگی ها از آنها با استفاده از الگوریتم های خوشه بندی مانند K-means و رگرسیون جنگل تصادفی است.

به طور خاص، از ابزارها و تکنیک‌های مختلف برای جمع‌آوری و تبدیل داده‌ها برای تولید ماشین‌های حالت محدود (FSM) و استخراج ویژگی‌ها از آنها برای شناسایی تاثیرگذارترین ویژگی‌های طراحی استفاده کردیم.

برای رسیدن به این اهداف، از ترکیبی از تکنیک های یادگیری ماشین و الگوریتم های خوشه بندی استفاده کردیم. از تکنیک های یادگیری ماشین برای تجزیه و تحلیل خطوط تولید نرم افزارهای دنیای واقعی و تولید مجموعه ای از FSM های نماینده استفاده کردیم. سپس، از الگوریتم‌های خوشه‌بندی مانند K-means و رگرسیون جنگل تصادفی برای استخراج ویژگی‌ها از هاFSM و شناسایی ویژگی‌های طراحی حیاتی که آنها را متمایز می‌کند،

استفاده کردیم. این ویژگی‌ها برای اعتبارسنجی ویژگی‌های طراحی حیاتی شناسایی‌شده و ارزیابی تأثیر آن‌ها بر تلاش تأیید مورد نیاز برای هاFSM در خطوط تولید نرم‌افزار استفاده می‌شوند.

از طریق این روش تحقیق، هدف ایجاد یک چارچوب جامع برای تولید FSM های تصادفی و استخراج ویژگی ها از آنها برای بهینه سازی فرآیند طراحی و آزمایش خطوط تولید نرم افزار است.

۶.۱ خلاصه فصل‌ها

در این فصل، مقدمه‌ای از نحوه‌ی سنتر خط محصول نرم‌افزاری برای ارزیابی تطبیقی یادگیری مدل و اهمیت این موضوع را بررسی کردیم. در فصل دوم به مرور ادبیات موضوع و کارهای انجام شده در گذشته می‌پردازیم و بیان میکنیم این تحقیق چگونه می‌تواند به تحقیق‌های گذشته کمک کند.

در فصل سوم، به توضیح روش استفاده شده در تحقیق میپردازیم. همچنین، ابزارهای استفاده شده برای جمع‌آوری و تغییر داده را برای ایجاد ماشین‌های حالت متناهی و انتخاب ویژگی‌هایی از آنها برای شناسایی تأثیرگذارترین آنها توضیح میدهیم. در فصل چهارم، به بررسی نتایج تحقیق و پاسخ دادن به مسائلی که مطرح کردیم میپردازیم. در پایان، تحلیل‌ها و نتایج بدست‌آمده بررسی و تحلیل می‌شوند و برخی مسائل آینده برای سایر محققان ذکر خواهد شد.

فصل ۲

مروری بر مطالعات انجام شده

۱.۲ مقدمه

در فصل پیشین مقدمه‌ای از نحوه‌ی سنتر خط محصول نرم‌افزاری برای ارزیابی تطبیقی یادگیری مدل و اهمیت این موضوع را بررسی کردیم. در این فصل به مرور ادبیات موضوع و کارهای انجام شده در گذشته می‌پردازیم و بیان میکنیم این تحقیق چگونه می‌تواند به تحقیق‌های گذشته کمک کند.

۲.۲ مروری بر ادبیات موضوع

در این بخش، تحقیق‌های پیشین در زمینه‌های مختلف مرتبط با موضوع را مرور می‌کنیم. در چند سال گذشته، علاقه فزاینده ای به توسعه سیستم های سازگار و هوشمندی که می توانند در طول زمان تکامل یابند، وجود داشته است. برای این منظور، محققان رویکردهای مختلفی را برای یادگیری و مدل‌سازی این سیستم‌ها بررسی کرده‌اند.

در این مقاله، ما یک بررسی جامع از سه مقاله که به این حوزه از تحقیقات کمک کرده‌اند، ارائه می‌کنیم. مطالعات پیشین یک رویکرد جدید برای یادگیری مدل تطبیقی [۱] برای سیستم‌های در حال تکامل [۲] پیشنهاد می‌کنند، که سیستم را قادر می‌سازد در حین استفاده مجدد از دانش موجود، یاد بگیرد و با داده‌های جدید سازگار

[1] Adaptive model learning
[2] Evolving systems

شود. همچنین، معیاری برای یادگیری فعال [3] سیستم‌های متغیر فشرده ارائه می‌دهند که یادگیری کارآمد و مؤثر سیستم‌های پیچیده با تنوع بالا را امکان‌پذیر می‌سازد. در نهایت، یک رویکرد یادگیری مدل رفتاری [4] تطبیقی را برای خطوط تولید نرم‌افزار معرفی می‌کنند، که سیستم را قادر می‌سازد تا ضمن حفظ رفتار موجود، یاد بگیرد و با ویژگی‌های جدید سازگار شود. از طریق تجزیه و تحلیل این مقالات، هدف ما ارائه یک درک جامع از وضعیت فعلی تحقیقات در سیستم‌های تطبیقی و هوشمند است که می‌تواند در طول زمان تکامل یابد و راه‌های بالقوه برای تحقیقات آینده در این زمینه را شناسایی کنیم.

- مقاله اول [۳] رویکرد جدیدی را برای یادگیری مدل تطبیقی پیشنهاد می‌کند که سیستم‌های در حال تکامل را قادر می‌سازد تا از داده‌های جدید بیاموزند در حالی که از دانش موجود استفاده مجدد می‌کنند. همچنین به چالش سیستم‌های در حال تکامل دائمی می‌پردازند، جایی که سیستم باید با محیط‌های در حال تغییر و داده‌های جدید سازگار شود. در ادامه، روشی را پیشنهاد می‌کنند که از یک مدل از پیش آموزش‌دیده همراه با تکنیک تقطیر دانش [5] استفاده می‌کند تا سیستم را قادر می‌سازد تا از داده‌های جدید در حین استفاده مجدد از دانش موجود بیاموزد. نتایج تجربی نشان می‌دهد که رویکرد آنها از روش‌های موجود بهتر عمل می‌کند و سیستم را قادر می‌سازد تا با حداقل آموزش اضافی، داده‌های جدید را یاد بگیرد و با آن‌ها سازگار شود.

  با استفاده از این مقاله، میتوان نقش حیاتی ماشین های حالت متناهی را در مدل سازی رفتار سیستم های نرم افزاری و توانمندسازی یادگیری مدل تطبیقی شناسایی کرد. ماشین های حالت متناهی می توانند به ویژه در مدل سازی سیستم های پیچیده با تنوع بالا، مانند خطوط تولید نرم افزار، مفید باشند. به طور خاص، این مقاله بر استفاده از ماشین های حالت متناهی به عنوان وسیله ای برای نمایش رفتار سیستم های نرم افزاری تأکید می کند. این بینش ها به تحقیقات ما در مورد اهمیت ماشین های حالت متناهی در مدل سازی رفتار خطوط تولید نرم افزار و پتانسیل یادگیری مدل تطبیقی برای بهبود عملکرد این سیستم ها کمک قابل توجهی کرد.

- مقاله دوم [۶] معیاری برای یادگیری فعال سیستم‌های پیچیده با تنوع بالا ارائه می‌کند. نویسندگان به چالش یادگیری از مجموعه داده های بزرگ و پیچیده می پردازند، جایی که سیستم باید آموزنده ترین

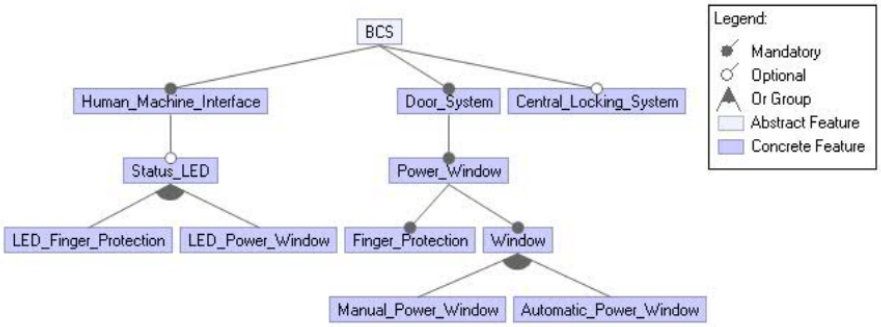
[3] Active learning
[4] Behavioral model learning
[5] Knowledge distillation technique

نمونه ها را برای برچسب گذاری انتخاب کند تا تلاش حاشیه نویسی را کاهش دهد و عملکرد یادگیری را بهبود بخشد. این معیار همچنین شامل چندین معیار ارزیابی مانند امتیاز F1، دقت و فراخوانی دقیق [۶] است. نویسندگان چندین الگوریتم یادگیری فعال را بر روی مجموعه داده‌های معیار ارزیابی می‌کنند و نشان می‌دهند که رویکرد پیشنهادی آنها از نظر عملکرد و کارایی از روش‌های موجود بهتر عمل می‌کند. این مقاله با برجسته کردن پتانسیل معیار آنها در تسهیل توسعه الگوریتم‌های یادگیری فعال جدید برای سیستم‌های متغیر فشرده به پایان می‌رسد.

- مقاله سوم [۵] رویکردی را برای یادگیری مدل رفتاری تطبیقی در خطوط تولید نرم افزار [۷] معرفی می کند و به چالش توسعه محصولات نرم افزاری با انواع و ویژگی های متعدد می پردازند، جایی که رفتار نرم افزار ممکن است بسته به ویژگی های انتخاب شده تغییر کند. همچنین، رویکردی را پیشنهاد می کنند که از بازخورد کاربران خط تولید برای یادگیری و تطبیق مدل های رفتاری [۸] نرم افزار استفاده می کند. سپس، رویکرد خود را در یک مطالعه موردی در صنعت خودرو نشان داده شده در تصویر ۱.۲ ارزیابی می‌کنند، جایی که خط تولید نرم‌افزار شامل انواع و ویژگی‌های متعدد است. نتایج تجربی نشان می‌دهد که رویکرد آنها نرم‌افزار را قادر می‌سازد تا با ویژگی‌های جدید سازگار شود و در عین حال رفتار موجود را حفظ کند، عملکرد کلی و رضایت کاربر را بهبود بخشد.

در این مقاله، نویسندگان بر اهمیت مدل‌های ویژگی [۹] در خطوط تولید نرم‌افزار تأکید می‌کنند، زیرا می‌توانند تأثیر قابل‌توجهی بر رفتار نرم‌افزار داشته باشند. در این مقاله اشاره شده است که خطوط تولید نرم‌افزار می توانند صدها یا حتی هزاران ویژگی داشته باشند که می تواند مدل سازی رفتاری این سیستم ها را چالش برانگیز کند. همچنین، رویکردی را برای یادگیری مدل رفتاری تطبیقی پیشنهاد می‌کند که تنوع ویژگی‌ها را در خطوط تولید نرم‌افزار در نظر می‌گیرد و سیستم را قادر می‌سازد تا با ویژگی‌های جدید و در عین حال حفظ رفتار موجود سازگار شود. به کمک این مقاله ویژگی‌های حیاتی خطوط تولید محصول را در تحقیقات خود شناسایی کردیم، که به تحقیقات آینده در مورد توسعه الگوریتم های یادگیری مدل موثر برای خطوط تولید محصول کمک می کند.

[6] Precision-recall
[7] Software product lines
[8] Behavioral models
[9] Feature models



شکل ۱.۲: مدل ویژگی یک نسخه ساده شده از خط تولید نرم‌افزار Body Comfort System [۴]

۳.۲ نتیجه‌گیری

در پایان، مقاله‌های بررسی شده، بینش های ارزشمندی را در زمینه سیستم های تطبیقی و هوشمند ارائه می دهند که می توانند در طول زمان تکامل یابند. این مطالعات رویکردها و معیارهای جدیدی را پیشنهاد می‌کنند که سیستم‌ها را قادر می‌سازد تا به طور کارآمد و مؤثر با داده‌ها و ویژگی‌های جدید یاد بگیرند و با آنها سازگار شوند. با استفاده از این رویکردها و معیارها برای سناریوهای دنیای واقعی، پتانسیل این تکنیک ها را در تسهیل توسعه سیستم های هوشمند و سازگار نشان داده میشود.

در تحقیقات خود، ما قصد داریم از این مطالعات به عنوان پایه ای برای بررسی ویژگی هایی استفاده کنیم که خطوط تولید نرم افزار را پیچیده ترین می کند. با تولید تصادفی خطوط تولید نرم‌افزار و مطالعه آنها، هدف ما شناسایی ویژگی های حیاتی است که خطوط تولید نرم‌افزارها را پیچیده و یادگیری را چالش برانگیز می کند. این تحقیق ما را قادر می سازد الگوریتم های یادگیری مدل [۱۰] خود را بهبود بخشیم و اطمینان حاصل کنیم که آنها به طور موثر و کارآمد در سناریوهای دنیای واقعی کار می کنند.

[10] Model learning algorithms

فصل ۳

روش تحقیق

۱.۳ مقدمه

در این فصل، به توضیح روش استفاده شده در تحقیق به طول کامل میپردازیم. در این بخش، ابزارهای استفاده شده نحوه جمع‌آوری و تغییر داده را برای ایجاد ماشین‌های حالت متناهی و انتخاب ویژگی‌ها از آنها برای شناسایی تأثیرگذارترین آنها توضیح داده‌ایم.

۲.۳ تشریح کامل روش تحقیق

این تحقیقات بر توسعه الگوریتم های یادگیری موثر برای خطوط تولید نرم افزار متمرکز است. به طور خاص، اجرای این کار شامل چندین مرحله کلیدی از جمله تولید FSM های تصادفی، استخراج ویژگی ها از آنها با استفاده از تکنیک های خودکار، و تجزیه و تحلیل داده های جمع آوری شده برای شناسایی حیاتی‌ترین ویژگی ها بود. با مطالعه این ویژگی‌ها، می‌توان الگوریتم‌های یادگیری مؤثرتری را توسعه داد، که می‌تواند کارایی و اثربخشی سیستم را در طول زمان بهبود بخشد.

۱.۲.۳ یادگیری مدل تطبیقی برای سیستم‌های در حال تکامل

اجرای رویکرد یادگیری مدل تطبیقی پیشنهادی شامل چندین مرحله کلیدی، از جمله پیش‌آموزش یک مدل، تولید مجموعه‌ای از مدل‌ها، و استفاده از تقطیر دانش [۱] برای انتقال دانش از مجموعه به مدلی دیگر است. ماشین های Mealy نوعی ماشین حالت متناهی هستند که هم حالت ها و هم خروجی ها را شامل می شود. از ماشین‌های Mealy میتوان به عنوان یک تکنیک مدل‌سازی برای نشان دادن رفتار سیستم و گرفتن رابطه بین ورودی‌ها و خروجی‌های سیستم استفاده کرد.

۲.۲.۳ تولید ماشین‌های حالت متناهی تصادفی

ماشین حالت محدود یک مدل ریاضی است که برای توصیف سیستم هایی استفاده می شود که می توانند در هر زمان معین در یکی از تعداد محدودی از حالت [۲] ها باشند. در زمینه کد ارائه شده، یک FSM به عنوان یک گراف جهت دار نشان داده می شود که در آن گره ها حالت ها را نشان می دهند و لبه ها [۳] انتقال [۴] بین حالت ها را نشان می دهند. هر انتقال با یک نماد ورودی و یک نماد خروجی برچسب‌گذاری می‌شود که به ترتیب نشان‌دهنده ورودی است که انتقال را راه‌اندازی می‌کند و خروجی‌ای که توسط انتقال تولید می‌شود. برای تولید انبوه این ماشین‌ها ما به صورت تصادفی ما از روش زیر استفاده کردیم. در اینجا یک تفکیک دقیق از کارهای انجام شده آورده شده است:

- ماژول‌ها و کتابخانه‌های ضروری شامل random، string، os، graphviz و tqdm هستند. random برای تولید اعداد تصادفی، string برای تولید رشته ها، OS برای تعامل با سیستم عامل، graphviz برای رندر کردن FSM ها، و tqdm برای نمایش پیشرفته استفاده می شود.
- یک کلاس اصلی برای یک ماشین حالت محدود تعریف می‌شود که دارای سه متغیر نمونه است: Isize تعداد ورودی‌ها، Ssize تعداد هاstate و name نشان دهنده نام FSM است. همچنین دارای یک متد generate states است که لیستی از نام‌های state را بر اساس مقدار Isize تولید می‌کند، یک متد

[1] Knowledge distillation
[2] State
[3] Edge
[4] Transition

generate random graph که یک گراف تصادفی برای FSM بر اساس مقادیر Ssize و Isize تولید می‌کند، و یک متد create fsm که FSM را در یک فایل متنی می‌نویسد. در نهایت، یک متد render fsm دارد که FSM را با استفاده از کتابخانه graphviz رندر کرده و نمایش می‌دهد.

- متد generate random graph یک نمودار تصادفی برای FSM تولید می کند. با ایجاد یک dictionary خالی برای نشان دادن نمودار شروع می شود. سپس فهرستی از نام‌های انتقال ممکن را با استفاده از متد generate states ایجاد می‌کند که فهرستی از رشته‌ها را بر اساس مقدار Isize ایجاد می‌کند. برای هر حالت در ،FSM تعداد تصادفی انتقال بین ۱ و Isize ایجاد می کند. برای هر انتقال، به طور تصادفی یک نام انتقال را از لیست نام های ممکن انتخاب می کند، یک مقدار خروجی تصادفی ۰ یا ۱ ایجاد می کند، و به طور تصادفی یک حالت هدف را از لیست حالت ها انتخاب می کند. سپس این انتقال را برای وضعیت فعلی به نمودار اضافه می کند.

- متد create fsm با ایجاد دایرکتوری به نام fsm در دایرکتوری فعلی کار شروع می شود. سپس با استفاده از روش generate random graph یک نمودار تصادفی برای FSM تولید می کند. برای هر حالت در FSM روی انتقال‌های آن حالت تکرار می‌شود و یک خط در فایل متنی با فرمت مشخص شده می‌نویسد. در ادامه FSM را در یک فایل متنی با فرمت زیر می نویسد:

state – transition/output → target_state

در اینجا معنای هر قسمت از خط آمده است:

- state : وضعیت فعلی .FSM
- transition : نام انتقالی که توسط نماد ورودی ایجاد می شود.
- output : خروجی حاصل از انتقال (۰ یا ۱).
- target_state : حالتی که FSM پس از راه اندازی انتقال به آن منتقل می شود.

در اینجا مثالی از نحوه ظاهر خروجی یک FSM ساده آورده شده است:

1 – c/1 → 2

$1 - e/1 \rightarrow 5$

$1 - d/1 \rightarrow 5$

$2 - e/0 \rightarrow 6$

$2 - c/0 \rightarrow 3$

$3 - b/0 \rightarrow 1$

$3 - c/1 \rightarrow 5$

$4 - b/1 \rightarrow 6$

$4 - a/0 \rightarrow 1$

$5 - e/0 \rightarrow 1$

$6 - b/1 \rightarrow 5$

$6 - c/0 \rightarrow 1$

$6 - d/0 \rightarrow 5$

$6 - e/0 \rightarrow 1$

$7 - a/1 \rightarrow 5$

$7 - b/0 \rightarrow 2$

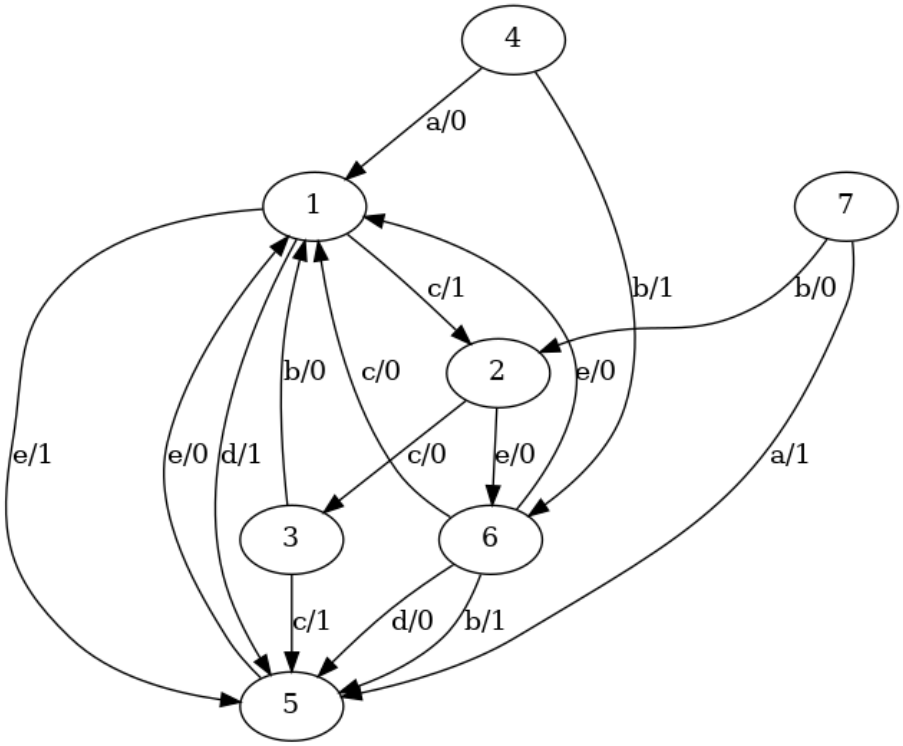
در این مثال، FSM دارای هفت حالت و پنج انتقال است. برای مثال، انتقال اول از حالت ۱ به حالت ۲ با ورودی a با خروجی ۱ می رود.

- متد render fsm از کتابخانه graphviz برای رندر FSM به عنوان تصویر PNG استفاده می‌کند و آن را در دایرکتوری fsm ذخیره می کند. با ایجاد یک Dot object شروع می‌شود، که برای نمایش FSM به عنوان یک گراف جهت‌دار استفاده می شود. سپس گره‌ها [۵] را برای هر حالت و یال ها [۶] را برای هر انتقال به نمودار اضافه می کند. در نهایت، نمودار را به صورت تصویر PNG ۱.۳ ارائه می کند و آن را در دایرکتوری fsm ذخیره می کند.

یکی از جنبه های مهم FSM ها توانایی آنها در مدل سازی سیستم های پیچیده با استفاده از اجزای نسبتا

[5] Node
[6] Edge

ساده است. با تجزیه یک سیستم به تعداد محدودی از حالت ها و انتقال ها، یک FSM می تواند توصیفی واضح و مختصر از رفتار سیستم ارائه دهد. این باعث می شود FSM ها برای مدل سازی و تجزیه و تحلیل طیف گسترده ای از سیستم ها، از جمله سیستم هایی با رفتار پیچیده و غیر خطی مفید باشند. به طور کلی، کد ارائه شده ابزار مفیدی برای تولید و تجسم FSM های تصادفی است که می تواند برای آزمایش و آزمایش در زمینه های مختلف استفاده شود.



شکل ۱.۳: نمونه‌ای از FSM های تصادفی تولید شده

۳.۲.۳ اجرای الگوریتم یادگیری مدل

در زمینه یادگیری ماشین‌ها [۷] ، فرآیند یادگیری شامل استنباط یک ماشین حالت متناهی (FSM) است که به طور دقیق رفتار یک سیستم ناشناخته را مدل می کند. هدف یادگیری ساختار زیربنایی سیستم از مجموعه نمونه های ورودی/خروجی ارائه شده توسط سیستم است. فرآیند یادگیری معمولاً شامل چندین دور تعامل بین یادگیرنده و سیستم است.

[7] Automata learning

در هر دور، یادگیرنده یک فرضیه FSM تولید می کند که رفتار مشاهده شده سیستم را توضیح می دهد. سپس فرضیه FSM با ارسال دنباله های ورودی و مقایسه خروجی فرضیه FSM با خروجی واقعی سیستم در برابر سیستم آزمایش می شود. اگر فرضیه FSM به درستی رفتار سیستم را مدل کند، فرآیند یادگیری به دور بعدی ادامه می‌یابد. در غیر این صورت، یادگیرنده بر اساس مغایرت های مشاهده شده، فرضیه FSM را اصلاح می کند و دور جدیدی را آغاز می کند.

فرآیند یادگیری را می توان به دو مرحله اصلی تقسیم کرد: مرحله یادگیری و مرحله آزمایش. در مرحله یادگیری، یادگیرنده برای ایجاد یک فرضیه FSM با سیستم تعامل دارد. در مرحله آزمون، یادگیرنده با ارسال توالی های ورودی و مقایسه خروجی فرضیه FSM با خروجی واقعی سیستم، صحت فرضیه FSM را ارزیابی می کند.

از مفاهیم اساسی در زمینه تئوری ماشین [۸] و یادگیری ماشین Membership query ها و Equivalence query [۲] ها هستند. Membership query درخواستی برای تعیین اینکه آیا یک دنباله ورودی داده شده توسط یک ماشین حالت محدود پذیرفته یا رد شده است. در مقابل، یک Equivalence query درخواستی برای مقایسه دو FSM و تعیین اینکه آیا آنها یک زبان را قبول دارند یا خیر است. این پرسش‌ها معمولاً در زمینه یادگیری خودکار استفاده می‌شوند، جایی که هدف استنباط FSM است که به درستی رفتار یک سیستم ناشناخته را مدل‌سازی می‌کند.

در زمینه کار ما، از Membership query و Equivalence query برای ارزیابی عملکرد الگوریتم های یادگیری مدل و تجزیه و تحلیل تأثیر ویژگی های مختلف FSM ها بر فرآیند یادگیری استفاده شد. تعداد بازنشانی‌های Membership query [۹] و Equivalence query برای نظارت بر پیشرفت فرآیند یادگیری و شناسایی تنگناهای بالقوه استفاده میشوند. با تجزیه و تحلیل این عوامل، هدف ما شناسایی فرصت‌هایی برای بهبود کارایی فرآیند یادگیری و توسعه الگوریتم‌های یادگیری مدل بهتر است که می‌تواند تعداد دورها و بازنشانی‌های مورد نیاز برای استنتاج دقیق مدل را کاهش دهد.

در مطالعه قبلی [۳] ، از الگوریتم Lstar [۱] برای یادگیری مدل استفاده شد. برای بررسی تأثیر ویژگی‌های مختلف ماشین‌های حالت محدود (FSM) بر فرآیند یادگیری، کد پایه را تغییر دادیم تا FSM های در حال اجرا را فعال کرده و تعداد دورها، بازنشانی‌های Membership query و بازنشانی‌های Equivalence query را ثبت کنیم. سپس این ویژگی ها برای شناسایی تأثیرگذارترین عوامل مورد تجزیه و تحلیل قرار گرفتند. هدف، تولید

[8] Automata theory
[9] Reset

FSM های پیچیده برای مطالعات بیشتر و ارزیابی عملکرد الگوریتم های یادگیری مدل در کاهش تعداد دورها و تنظیم مجدد، در مقایسه با الگوریتم Lstar بود.

فرآیند یادگیری می تواند از نظر محاسباتی گران باشد، به ویژه برای سیستم های پیچیده با فضاهای ورودی/خروجی بزرگ. بنابراین، توسعه الگوریتم‌های یادگیری مدل کارآمد که می‌تواند تعداد دورها و پرس‌وجوهای مورد نیاز برای استنتاج مدل دقیق را به حداقل برساند، یک حوزه مهم تحقیق در یادگیری ماشین است.

۴.۲.۳ استخراج ویژگی‌ها

پس از تولید مجموعه‌ای از ماشین‌های حالت متناهی و استفاده از آن‌ها برای ارزیابی عملکرد الگوریتم یادگیری خود، به استخراج ویژگی‌هایی پرداختیم که می‌توانند بر فرآیند یادگیری تأثیر بگذارند. به طور خاص، ما تعداد دورهای مورد نیاز توسط الگوریتم را برای یادگیری دقیق هاFSM تجزیه و تحلیل کردیم. ما فرض کردیم که ویژگی‌های خاصی از ،هاFSM مانند اندازه، پیچیدگی، و رفتار ورودی/خروجی آن‌ها، می‌تواند تأثیر قابل‌توجهی بر فرآیند یادگیری داشته باشد. برای آزمایش این فرضیه، ما یک تحلیل ویژگی [۱۰] را با محاسبه همبستگی بین دوره‌های یادگیری و ویژگی‌های مختلف هاFSM انجام دادیم. ما چندین ویژگی را شناسایی کردیم که با تعداد دورها همبستگی داشتند و از این اطلاعات برای ایجاد مجموعه‌ای از دستورالعمل‌ها برای طراحی FSM هایی استفاده کردیم که برای یادگیری کارآمدتر سازگار هستند.

۱.۴.۲.۳ درجه همبندی یال‌های جهت دار

در تئوری گراف، "درجه همبندی یال‌های جهت‌دار" [۱۱] به حداقل تعداد یال های جهت دار اطلاق می شود که باید از یک گراف جهت دار حذف شوند تا اتصال آن قطع شود. به عبارت دیگر، معیاری برای استحکام اتصال گراف در حضور شکست یال ها است.

مفهوم درجه همبندی یال‌های جهت‌دار ارتباط نزدیکی با همبندی راس دارد، که حداقل تعداد راس هایی را که باید از یک گراف غیر جهت دار حذف شوند تا قطع شود، اندازه گیری می کند. با این حال، در گراف‌های جهت دار، مفهوم همبندی راس به خوبی تعریف نشده است، زیرا حذف یک راس ممکن است لزوماً گراف را قطع نکند.

[10] Feature analysis
[11] Arc connectivity

بنابراین، درجه همبندی یال‌های جهت‌دار اغلب به عنوان یک معیار جایگزین برای اتصال در نمودارهای جهت دار استفاده می شود.

درجه همبندی یال‌های جهت‌دار یک ویژگی مفید برای شناسایی مهم ترین یال‌های جهت‌دار در یک ماشین حالت متناهی و برای انتخاب بهترین مدل مناسب برای یک مجموعه داده معین است. با محاسبه درجه همبندی یال‌های جهت‌دار FSM های مختلف، می توانیم تعیین کنیم که کدام مدل ها در برابر خرابی یال‌ها مقاوم‌تر و انعطاف پذیرتر هستند و بنابراین احتمال بیشتری دارد که رفتار زیربنایی سیستم را به دقت ثبت کنند.

علاوه بر این، اتصال قوس می‌تواند برای بهینه‌سازی فرآیند یادگیری با هدایت انتخاب توالی‌های ورودی/خروجی که به احتمال زیاد ویژگی‌های حیاتی FSM را آشکار می‌کنند، استفاده شود. با تمرکز بر روی مهمترین یال ها، می توانیم ابعاد فضای جستجو را کاهش دهیم و کارایی الگوریتم یادگیری را بهبود بخشیم. این می تواند به ویژه برای FSM های بزرگ و پیچیده مفید باشد، جایی که تعداد توالی های ورودی/خروجی ممکن به شدت زیاد است.

به طور خلاصه، ترکیب اتصال قوس به عنوان یک ویژگی در فرآیند یادگیری مدل می تواند به شناسایی مناسب ترین FSM ها برای یک مجموعه داده معین و بهبود کارایی الگوریتم یادگیری کمک کند.

برای محاسبه درجه همبندی یال‌های جهت‌دار یک ماشین حالت متناهی، از الگوریتم Ford-Fulkerson به همراه کتابخانه NetworkX در پایتون استفاده کردیم. الگوریتم Ford-Fulkerson یک الگوریتم کلاسیک برای محاسبه حداکثر جریان در یک شبکه است. در این حالت، از آن برای محاسبه حداکثر تعداد مسیرهای ناهمگون یال از حالت اولیه تا حالت نهایی FSM استفاده کردیم.

FSM را به عنوان یک گراف جهت دار با استفاده از ساختار داده dictionary نشان دادیم، جایی که هر کلید یک حالت را نشان می دهد و هر مقدار فهرستی از یال های خروجی را نشان می دهد.

برای محاسبه درجه همبندی یال‌های جهت‌دار، گراف ابتدا به یک شی NetworkX DiGraph تبدیل می شود. سپس یک گره مبدا و یک گره نهایی به گراف اضافه می شوند تا به ترتیب حالت های اولیه و نهایی را نشان دهند. گره مبدا به همه حالت ها به جز حالت اولیه و حالت نهایی به گره نهایی متصل است. به هر یال ظرفیت ۱ و جریان ۰ اختصاص داده شده است. سپس مقدار حداکثر جریان محاسبه می شود. حداکثر جریان نشان دهنده حداکثر تعداد مسیرهای پراکنده یال از حالت اولیه تا حالت نهایی است.

در نهایت، درجه همبندی یال‌های جهت‌دار به عنوان تعداد یال‌ها در dictionary جریان منهای ۲ محاسبه می‌شود. این مقدار نشان دهنده حداقل تعداد یالی است که باید از نمودار حذف شوند تا حالت اولیه و نهایی قطع شود.

۲.۴.۲.۳ نسبت ورودی ها به تعداد حالت ها

نسبت ورودی‌ها به تعداد حالت‌ها [۱۲] پارامتری است که معمولاً در تجزیه و تحلیل ماشین‌های حالت متناهی استفاده می‌شود.نسبت ورودی به حالت معیار مهمی از پیچیدگی یک FSM است. به طور کلی، نسبت ورودی به حالت بالاتر نشان‌دهنده یک FSM پیچیده‌تر است، زیرا به این معنی است که هر حالت باید بتواند تعداد بیشتری از ترکیب‌های ورودی را مدیریت کند. برعکس، نسبت ورودی به حالت پایین‌تر نشان‌دهنده یک FSM ساده‌تر است، زیرا هر حالت باید ترکیب‌های ورودی کمتری را مدیریت کند.

برای مثال، نسبت ورودی به حالت یکی از چندین عاملی است که می تواند بر طراحی و بهینه سازی مدارهای دیجیتال تأثیر بگذارد. برای مثال، نسبت ورودی به حالت بالا ممکن است به مدارهای پیچیده تری برای پیاده سازی FSM نیاز داشته باشد، در حالی که نسبت ورودی به حالت پایین ممکن است منجر به یک مدار کوچکتر، سریعتر، اما کمتر انعطاف پذیر شود.

۳.۴.۲.۳ میانگین طول مسیر

میانگین طول مسیر یک ویژگی مفید برای توصیف پیچیدگی و کارایی یک ماشین حالت متناهی است. میانگین طول کوتاه ترین مسیرها را بین تمام جفت حالت ها در FSM اندازه گیری می کند و می تواند برای مقایسه ویژگی های ساختاری FSM های مختلف استفاده شود. برای محاسبه میانگین طول مسیر، از کتابخانه NetworkX در پایتون برای محاسبه کوتاه ترین طول مسیر بین تمام جفت حالت ها در FSM استفاده می کند. الگوریتم به صورت زیر کار میکند:

ابتدا FSM به یک شی NetworkX DiGraph تبدیل می شود. سپس، یک dictionary ایجاد می شود تا کوتاه ترین طول مسیرها را بین تمام جفت گره ها ذخیره کند. کوتاه ترین طول مسیر با استفاده از کتابخانه NetworkX محاسبه می شود. در مرحله بعد، میانگین طول مسیر با جمع کردن کوتاه ترین طول مسیر بین تمام جفت گره ها و تقسیم بر تعداد کل جفت ها محاسبه می شود. اگر کوتاه ترین مسیر بین دو گره وجود نداشته باشد، طول مسیر بی نهایت در نظر گرفته می شود. مقدار به دست آمده نشان دهنده طول متوسط کوتاه ترین مسیرها بین تمام جفت حالت ها در FSM است.

میانگین طول مسیر می تواند یک ویژگی مفید برای شناسایی FSM هایی باشد که کم و بیش از سایرین پیچیده

[12] Input-to-state ratio

هستند. میانگین طول مسیر بالاتر نشان‌دهنده یک FSM پیچیده‌تر با مسیرهای طولانی‌تر بین حالت‌ها است، در حالی که میانگین طول مسیر کمتر نشان‌دهنده یک FSM ساده‌تر با مسیرهای کوتاه‌تر بین حالت‌ها است. این اطلاعات می تواند برای هدایت انتخاب الگوریتم های یادگیری مناسب و بهینه سازی طراحی FSM ها برای کاربردهای خاص مورد استفاده قرار گیرد.

به طور خلاصه، کد بالا از کتابخانه NetworkX برای محاسبه میانگین طول مسیر یک FSM معین استفاده می کند. این الگوریتم با محاسبه کوتاه ترین طول مسیر بین تمام جفت حالت ها و سپس محاسبه میانگین آن طول ها کار می کند. مقدار به دست آمده را می توان به عنوان یک ویژگی برای مقایسه ویژگی های ساختاری FSM های مختلف و برای بهینه سازی طراحی آن‌ها برای کاربردهای خاص استفاده کرد.

۴.۴.۲.۳ فرکانس انتقال حالت

فرکانس انتقال حالت [۱۳] یک ویژگی مفید برای توصیف فرکانس انتقال حالت در یک ماشین حالت متناهی (FSM) است. تعداد کل دفعاتی که هر انتقال حالت در FSM رخ می دهد را اندازه گیری می کند و می تواند برای شناسایی متداول ترین یا مهم ترین انتقال ها در سیستم استفاده شود. برای محاسبه فرکانس انتقال حالت، کد بالا تعداد دفعاتی را که هر انتقال حالت در FSM رخ می دهد، می شمارد. الگوریتم به صورت زیر کار میکند:

اول، FSM به عنوان یک ساختار داده ی dictionary نشان داده می شود، که در آن هر کلید نشان دهنده یک انتقال حالت و هر مقدار نشان دهنده لیستی از نمادهای ورودی/خروجی مرتبط با آن انتقال است. سپس، الگوریتم بر روی تمام انتقال‌های حالت ممکن در FSM تکرار می‌شود و فرکانس هر گذار را با جمع کردن تعداد جفت‌های نماد ورودی/خروجی مرتبط با آن انتقال، شمارش می‌کند. مقدار حاصل تعداد کل دفعاتی را نشان می دهد که هر انتقال حالت در FSM رخ می دهد و می تواند به عنوان یک ویژگی برای مشخص کردن فرکانس و اهمیت انتقال حالت های مختلف در سیستم استفاده شود.

فرکانس انتقال حالت می تواند یک ویژگی مفید برای شناسایی مهم ترین یا متداول ترین انتقال ها در FSM باشد. این اطلاعات می تواند برای هدایت انتخاب الگوریتم های یادگیری مناسب و بهینه سازی طراحی FSM ها برای کاربردهای خاص مورد استفاده قرار گیرد.

[13] State transition frequency

۵.۴.۲.۳ پوشش حالت

پوشش حالت [۱۴] یک ویژگی مفید برای توصیف جامعیت یک ماشین حالت متناهی در نمایش رفتار یک سیستم است. پوشش حالت درصد حالت‌هایی را در FSM اندازه‌گیری می‌کند که حداقل یک دنباله ورودی از آنها بازدید می‌شود و می‌تواند برای شناسایی کامل بودن و دقت FSM در ثبت رفتار سیستم استفاده شود.

برای محاسبه پوشش حالت، از کتابخانه NetworkX در پایتون برای محاسبه کوتاه‌ترین طول مسیر از هر حالت در FSM به همه حالت‌های دیگر استفاده می‌کند. ابتدا FSM به یک شی DiGraph NetworkX تبدیل می شود. سپس، الگوریتم بر روی تمام حالت‌های ممکن در FSM تکرار می‌شود و مجموعه حالت‌های بازدید شده را با یافتن کوتاه‌ترین طول مسیر از آن حالت تا همه حالت‌های دیگر در FSM محاسبه می‌کند. مجموعه به دست آمده حالت هایی را نشان می دهد که حداقل یک دنباله ورودی از آنها بازدید می کند.

در نهایت، پوشش حالت با تقسیم تعداد حالت‌های بازدید شده بر تعداد کل حالت‌ها در FSM محاسبه می شود. مقدار به دست آمده نشان دهنده درصد حالت هایی در FSM است که حداقل یک دنباله ورودی از آنها بازدید می کند و می تواند به عنوان یک ویژگی برای توصیف جامعیت و دقت FSM در نمایش رفتار سیستم استفاده شود.

پوشش حالت می تواند یک ویژگی مفید برای شناسایی کامل بودن و دقت FSM در نمایش رفتار سیستم باشد. پوشش حالت بیشتر نشان‌دهنده FSM جامع‌تر و دقیق‌تر است که درصد بیشتری از رفتار سیستم را ثبت می‌کند، در حالی که پوشش وضعیت کمتر نشان‌دهنده FSM کمتر جامع و دقیق‌تر است که درصد کمتری از رفتار سیستم را نشان می‌دهد. این اطلاعات می تواند برای هدایت انتخاب الگوریتم های یادگیری مناسب و بهینه سازی طراحی FSM ها برای کاربردهای خاص مورد استفاده قرار گیرد.

۶.۴.۲.۳ آنتروپی حالت

آنتروپی حالت [۱۵] یک ویژگی مفید برای مشخص کردن عدم قطعیت یا تصادفی یک ماشین حالت متناهی در نمایش رفتار یک سیستم است. آنتروپی سطح عدم قطعیت یا تصادفی بودن توزیع حالت های بازدید شده توسط FSM را اندازه گیری می کند و می تواند برای شناسایی پیچیدگی و تنوع رفتار سیستم استفاده شود.

[14] State coverage
[15] State entropy

برای محاسبه آنتروپی حالت، کتابخانه NetworkX و کتابخانه NumPy در پایتون برای محاسبه آنتروپی توزیع حالت استفاده می کند. ابتدا FSM به یک شی DiGraph NetworkX تبدیل می شود. سپس، الگوریتم بر روی همه حالت‌های ممکن در FSM تکرار می‌شود و فرکانس هر حالت را با یافتن کوتاه‌ترین طول مسیر از آن حالت تا همه حالت‌های دیگر در FSM محاسبه می‌کند. لیست به دست آمده نشان دهنده بسامد هر حالت در FSM است. در مرحله بعد، فرکانس های حالت با تقسیم هر فرکانس بر تعداد کل حالت های بازدید شده به احتمالات نرمال می شوند. احتمالات به دست آمده نشان دهنده احتمال بازدید از هر حالت توسط FSM است.

در نهایت، آنتروپی حالت با اعمال فرمول آنتروپی برای توزیع احتمال حالت ها محاسبه می شود. مقدار به دست آمده نشان دهنده سطح عدم قطعیت یا تصادفی در توزیع حالت های بازدید شده توسط FSM است و می تواند به عنوان ویژگی برای توصیف پیچیدگی و تنوع رفتار سیستم استفاده شود.

آنتروپی حالت می تواند یک ویژگی مفید برای شناسایی پیچیدگی و تنوع رفتار سیستم باشد. آنتروپی حالت بالاتر نشان‌دهنده رفتار سیستمی متنوع‌تر و پیچیده‌تر با سطح عدم قطعیت یا تصادفی بیشتر در توزیع حالت‌های بازدید شده توسط FSM است، در حالی که آنتروپی حالت پایین‌تر نشان‌دهنده رفتار سیستم کمتر متنوع و پیچیده‌تر با سطح پایین‌تر عدم قطعیت یا عدم اطمینان است. تصادفی در توزیع ایالات بازدید شده توسط .FSM این اطلاعات می تواند برای هدایت انتخاب الگوریتم های یادگیری مناسب و بهینه سازی طراحی FSM ها برای کاربردهای خاص مورد استفاده قرار گیرد.

۷.۴.۲.۳ پیچیدگی حالت

پیچیدگی حالت [۱۶] یک ویژگی مفید برای توصیف پیچیدگی یک ماشین حالت متناهی در نمایش رفتار یک سیستم است. سطح پیچیدگی هر حالت را در FSM اندازه گیری می کند و می تواند برای شناسایی سطح تلاش مورد نیاز برای درک و تحلیل رفتار سیستم استفاده شود.

برای محاسبه پیچیدگی حالت، کتابخانه NetworkX و کتابخانه NumPy در پایتون برای محاسبه پیچیدگی هر حالت در FSM استفاده می کند. ابتدا FSM به یک شی NetworkX DiGraph تبدیل می شود. سپس، الگوریتم بر روی تمام حالت‌های ممکن در FSM تکرار می‌شود و پیچیدگی هر حالت را با یافتن حداکثر کوتاه‌ترین طول مسیر از آن حالت تا همه حالت‌های دیگر در FSM محاسبه می‌کند. لیست به دست آمده نشان دهنده

[16] State complexity

پیچیدگی هر حالت در FSM است.

در نهایت، میانگین پیچیدگی حالت با در نظر گرفتن میانگین مقادیر پیچیدگی برای همه حالت‌ها در FSM محاسبه می‌شود. مقدار به دست آمده نشان دهنده سطح متوسط پیچیدگی حالت ها در FSM است و می تواند به عنوان ویژگی برای توصیف پیچیدگی رفتار سیستم استفاده شود.

پیچیدگی حالت می تواند یک ویژگی مفید برای شناسایی سطح تلاش مورد نیاز برای درک و تجزیه و تحلیل رفتار سیستم باشد. پیچیدگی حالت بالاتر نشان‌دهنده یک رفتار سیستم پیچیده‌تر است که ممکن است به تلاش بیشتری برای تجزیه و تحلیل و درک نیاز داشته باشد، در حالی که پیچیدگی حالت پایین‌تر نشان‌دهنده یک رفتار سیستم ساده‌تر است که ممکن است تجزیه و تحلیل و درک آسان‌تر باشد. این اطلاعات می تواند برای هدایت انتخاب الگوریتم های یادگیری مناسب و بهینه سازی طراحی FSM ها برای کاربردهای خاص مورد استفاده قرار گیرد.

۸.۴.۲.۳ پوشش انتقال

پوشش انتقال [۱۷] یک ویژگی مفید برای توصیف جامعیت یک ماشین حالت متناهی در نمایش رفتار یک سیستم است. درصد انتقال در FSM را که حداقل توسط یک دنباله ورودی انجام می شود اندازه گیری می کند و می تواند برای شناسایی کامل بودن و دقت FSM در ثبت رفتار سیستم استفاده شود.

برای محاسبه پوشش انتقال، از کتابخانه NetworkX در پایتون برای محاسبه درصد انتقال‌های گرفته شده استفاده می‌کنیم. ابتدا FSM به یک شی NetworkX DiGraph تبدیل می شود. سپس، الگوریتم تعداد کل انتقال‌ها در FSM و تعداد انتقال‌های گرفته شده را با تکرار روی تمام حالت‌های ممکن در FSM و شمارش تعداد یال‌های خروجی از هر حالت محاسبه می‌کند.

در نهایت، پوشش انتقال با تقسیم تعداد انتقال‌های گرفته شده بر تعداد کل انتقال‌ها در FSM محاسبه می‌شود. مقدار به دست آمده نشان دهنده درصد انتقال در FSM است که توسط حداقل یک دنباله ورودی گرفته شده است و می تواند به عنوان ویژگی برای توصیف جامعیت و دقت FSM در نمایش رفتار سیستم استفاده شود.

پوشش انتقال می تواند یک ویژگی مفید برای شناسایی کامل بودن و دقت FSM در نمایش رفتار سیستم باشد. پوشش انتقال بالاتر نشان‌دهنده یک FSM جامع‌تر و دقیق‌تر است که درصد بیشتری از رفتار سیستم را

[17] Transition coverage

نشان می‌دهد، در حالی که پوشش انتقال کمتر نشان‌دهنده یک FSM کمتر جامع و دقیق است که درصد کمتری از رفتار سیستم را نشان می‌دهد. این اطلاعات می تواند برای هدایت انتخاب الگوریتم های یادگیری مناسب و بهینه سازی طراحی FSM ها برای کاربردهای خاص مورد استفاده قرار گیرد.

۵.۲.۳ تحلیل داده‌ها

پس از استخراج ویژگی‌های ماشین‌های حالت متناهی، گام بعدی انجام تجزیه و تحلیل داده‌ها بر روی آنها برای شناسایی ویژگی‌هایی بود که بیشترین تأثیر را بر پیچیدگی هاFSM دارند. این تحلیل برای به دست آوردن بینشی در مورد روابط بین ویژگی‌ها و پیچیدگی هاFSM و شناسایی مهم‌ترین ویژگی‌ها برای مدل‌سازی و بهینه‌سازی انجام شد.

تجزیه و تحلیل داده ها با استفاده از تکنیک های آماری و یادگیری ماشینی، مانند تحلیل همبستگی [۱۸] ، انتخاب ویژگی [۱۹] ۲.۳ و مدل سازی انجام شد. تحلیل همبستگی برای شناسایی روابط زوجی بین ویژگی‌ها و پیچیدگی هاFSM استفاده شد، در حالی که انتخاب ویژگی برای شناسایی زیرمجموعه ویژگی‌هایی که بیشترین تأثیر را بر پیچیدگی هاFSM دارند، استفاده شد.

بینش به دست آمده از تجزیه و تحلیل داده ها می تواند برای هدایت انتخاب الگوریتم های یادگیری مناسب و بهینه سازی طراحی FSM ها برای کاربردهای خاص مورد استفاده قرار گیرد. به عنوان مثال، اگر مشخص شود که ویژگی‌های آنتروپی حالت برای یک برنامه خاص مهم‌ترین هستند، ممکن است به یک الگوریتم یادگیری که بتواند به طور موثر تنوع و پیچیدگی هاFSM را ضبط و مدل‌سازی کند، مورد نیاز است. از سوی دیگر، اگر مشخص شود که ویژگی پوشش انتقال برای یک برنامه خاص مهم‌تر است، یک الگوریتم یادگیری ساده‌تر ممکن است کافی باشد.

به طور کلی، تجزیه و تحلیل داده ها بینش های ارزشمندی را در مورد روابط بین ویژگی ها و پیچیدگی FSM ها ارائه کرد و مهم ترین ویژگی ها را برای مدل سازی و بهینه سازی شناسایی کرد. این بینش ها را می توان برای هدایت طراحی و توسعه FSM ها برای طیف گسترده ای از کاربردها، از جمله اتوماسیون استفاده کرد.

[18] Correlation analysis
[19] Feature selection

	SUL Name	Num States	Num Inputs	Self Loops	All Paths	Arch Connectivity	Average Path Length	State Transition Frequency	State Coverage	State Entropy	State Complexity	Transition Coverage	Input-to-state Ratio	Rounds
0	fsm_0	9	42	0.666667	186.888889	51	42.0	0	1.214286	5.304687	1.0	1.0	4.666667	3
1	fsm_1	5	13	0.400000	25.000000	18	13.0	0	1.384615	3.666390	1.0	1.0	2.600000	1
2	fsm_2	15	24	0.533333	35.400000	39	24.0	0	1.625000	4.459992	1.0	1.0	1.600000	1
3	fsm_3	7	31	0.714286	116.571429	38	31.0	0	1.225806	4.895444	1.0	1.0	4.428571	3
4	fsm_4	11	17	0.090909	26.272727	28	17.0	0	1.647059	4.007905	1.0	1.0	1.545455	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	fsm_995	9	7	0.666667	5.444444	16	7.0	0	2.285714	2.709983	1.0	1.0	0.777778	1
996	fsm_996	8	26	0.750000	81.375000	34	26.0	0	1.307692	4.600507	1.0	1.0	3.250000	1
997	fsm_997	16	19	0.437500	22.562500	35	19.0	0	1.842105	4.110737	1.0	1.0	1.187500	1
998	fsm_998	14	14	0.214286	14.000000	28	14.0	0	2.000000	3.631357	1.0	1.0	1.000000	1
999	fsm_999	2	4	0.000000	6.500000	6	4.0	0	1.500000	1.974938	1.0	1.0	2.000000	1

شکل ۲.۳: نمونه‌ای از ویژگی‌های انتخاب شده برای تحلیل‌داده

۱.۵.۲.۳ تحلیل همبستگی

برای محاسبه همبستگی، همبستگی زوجی بین تمام ستون‌های DataFrame را محاسبه می‌کنیم. ماتریس همبستگی حاصل یک ماتریس مربع است که در آن عناصر قطری ۱ هستند (زیرا یک متغیر کاملاً با خودش همبستگی دارد) و عناصر خارج از مورب ضرایب همبستگی بین جفت متغیرها هستند. ضریب همبستگی معیاری از رابطه خطی بین دو متغیر است و می تواند از −۱ (همبستگی منفی کامل) تا ۱ (همبستگی مثبت کامل) متغیر باشد که ۰ نشان دهنده عدم وجود همبستگی است.

برای انتخاب بهترین ویژگی‌های که در پیچیدگی FSM تاثیرگذارند، همبستگی بین ستون 'Rounds' و تمام ستون های دیگر در DataFrame را محاسبه می کند. با بررسی همبستگی بین ستون "Rounds" و تمام ستون های دیگر در ،DataFrame می توانیم بینش هایی در مورد روابط بین این متغیرها و ستون "Rounds" بدست آوریم. به عنوان مثال، اگر متغیری دارای همبستگی مثبت بالایی با ستون 'Rounds' باشد، نشان می دهد که با افزایش مقدار آن متغیر، تعداد دورها نیز تمایل به افزایش دارد. از سوی دیگر، اگر متغیری با ستون 'Rounds' همبستگی منفی بالایی داشته باشد، نشان می دهد که با افزایش مقدار آن متغیر، تعداد دورها کاهش می یابد.

به طور کلی، محاسبات همبستگی بین متغیرها یک تکنیک مفید برای شناسایی الگوها و روابط در داده ها است و می تواند برای هدایت انتخاب مدل های یادگیری ماشین مناسب و تکنیک های مهندسی ویژگی استفاده شود.

۲.۵.۲.۳ خوشه‌بندی

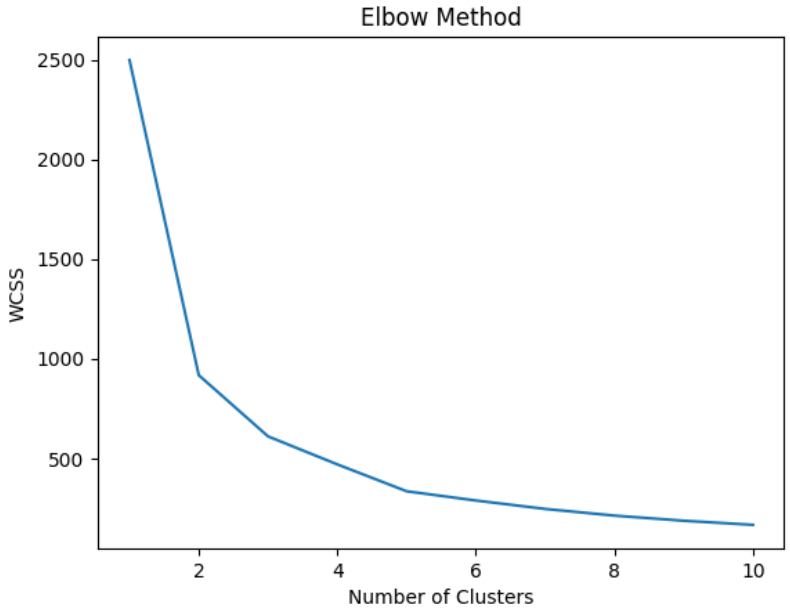
در ادامه، با استفاده از الگوریتم K-Means در حال انجام تجزیه و تحلیل خوشه بندی بر روی داده های DataFrame هستیم. هدف از خوشه‌بندی این است که نقاط داده مشابه را بر اساس ویژگی‌های آنها گروه‌بندی

کنیم، بدون اینکه دانش قبلی در مورد اینکه کدام نقاط داده در کدام گروه قرار دارند.

در ابتدا، مجموع مربع های درون خوشه ای (WCSS) را برای تعداد خوشه های مختلف با استفاده از روش elbow محاسبه می کند. WCSS اندازه گیری است که نشان می دهد نقاط داده در یک خوشه چقدر از مرکز خوشه پخش می شوند. روش elbow شامل ترسیم تعداد خوشه ها در برابر ،WCSS و انتخاب تعداد خوشه هایی است که کاهش WCSS شروع به تراز کردن می کند. ۳.۳ این به عنوان یک مبادله خوب بین تعداد خوشه ها و کیفیت خوشه بندی در نظر گرفته می شود.

بخش دوم کد از خوشه بندی K-Means برای گروه بندی نقاط داده در دو خوشه، بر اساس تعداد بهینه خوشه های شناسایی شده با استفاده از روش elbow استفاده می کند. همچنین از الگوریتم روش مقداردهی اولیه K-Means++ استفاده میکند که روش کارآمدتر و دقیق تری برای انتخاب مرکزهای خوشه Centroid اولیه است. آرگومان random_state=42 دانه تصادفی را برای تکرارپذیری تنظیم می کند. در ادامه هر نقطه داده را بر اساس نزدیکی آن به مرکز هر خوشه به یک خوشه اختصاص می‌دهد.

در انتها، برای رسم توزیع ستون 'Rounds' برای هر خوشه، از نمودار ویولن استفاده می شود. این رابطه بین برچسب های خوشه و ستون 'Rounds' را تجسم می کند و می تواند به شناسایی هر گونه الگو یا تفاوت بین خوشه ها کمک کند.



شکل ۳.۳: تعداد بهینه خوشه ها با استفاده از روش elbow

در ادامه، از خوشه بندی K-Means برای گروه بندی داده ها در DataFrame به دو خوشه استفاده می کنیم و سپس مرکز هر خوشه را محاسبه می کنیم. مرکزها مقادیر میانگین ویژگی های هر خوشه هستند و می توانند برای تفسیر ویژگی های هر خوشه استفاده شوند. به طور کلی، با محاسبه مرکزهای هر خوشه و مقادیر ویژگی هایی را که برای تمایز بین خوشه ها مهم هستند، می توانیم ویژگی های خوشه های مختلف را درک کرده و هر گونه الگو یا رابطه بین ویژگی ها و برچسب های خوشه شناسایی کنیم.

۳.۵.۲.۳ مدل رگرسیون جنگل تصادفی

رگرسیون جنگل تصادفی [۲۰] یک الگوریتم یادگیری ماشین است که چندین درخت تصمیم را برای ایجاد یک مدل پیش‌بینی دقیق‌تر ترکیب می‌کند. هر درخت تصمیم بر روی یک زیرمجموعه تصادفی از داده ها و یک زیرمجموعه تصادفی از ویژگی ها آموزش داده می شود که به کاهش بیش از حد برازش و افزایش توانایی تعمیم مدل کمک می کند. در طول پیش‌بینی، الگوریتم میانگین پیش‌بینی‌ها را از تمام درخت‌های تصمیم می‌گیرد تا پیش‌بینی نهایی را انجام دهد. رگرسیون جنگل تصادفی یک الگوریتم قدرتمند و انعطاف پذیر است که می تواند برای طیف گسترده ای از مسائل رگرسیون استفاده شود و تجزیه و تحلیل اهمیت ویژگی آن می تواند بینش های ارزشمندی را در مورد روابط بین متغیرها در یک مجموعه داده ارائه دهد.

ما از یک مدل رگرسیون جنگل تصادفی برای تعیین اهمیت ویژگی متغیرها در DataFrame استفاده می کنیم. هدف تجزیه و تحلیل اهمیت ویژگی، شناسایی متغیرهایی است که برای پیش‌بینی متغیر هدف، که در این مورد ستون «Rounds» است، مهم‌تر هستند. آرایه اهمیت به دست آمده حاوی اهمیت نسبی هر ویژگی است، جایی که مقدار بالاتر نشان دهنده اهمیت بیشتر است. این تکنیک می‌تواند برای انتخاب ویژگی، شناسایی پیش‌بینی‌کننده‌های مهم در یک مجموعه داده، و به دست آوردن بینش در مورد روابط بین متغیرها مفید باشد.

[20] Random forest regression

فصل ۴

نتایج

۱.۴ مقدمه

تا اینجا، در فصل اول، مقدمه‌ای از خطوط تولید نرم‌افزار را بررسی کردیم. در فصل دوم ادبیات تحقیق مورد بررسی قرار گرفت. در ادامه، روش جمع‌آوری داده و انتخاب ویژگی‌ها بررسی شد. در این فصل، به بررسی نتایج تحقیق و پاسخ دادن به مسائلی که در ابتدا مطرح شد میپردازیم. ابتدا با توجه به میزان همبستگی ویژگی‌ها با تعداد دورها ویژگی‌هایی که به پیچیده شدن FSM کمک میکنند را انتخاب میکنیم. در ادامه، به نتایج خوشه‌بندی میپردازیم.

۲.۴ بررسی ویژگی‌ها با توجه به میزان همبستگی

در این مقاله، استفاده از تکنیک‌های استخراج ویژگی و پیش پردازش برای تجزیه و تحلیل داده‌های ماشین حالت محدود (FSM) را بررسی کردیم. در بخش قبل، مجموعه‌ای از ویژگی‌ها را از داده‌های FSM استخراج کردیم، از جمله ویژگی‌های مربوط به تعداد حالت‌ها، انتقال‌ها و نمادهای ورودی/خروجی. سپس برخی از پیش‌پردازش‌ها را روی ویژگی‌ها انجام دادیم، مانند نرمال‌سازی، برای اطمینان از اینکه همه ویژگی‌ها مقیاس یکسانی دارند، و حذف ویژگی‌هایی که برای همه هاFSM ارزش یکسانی داشتند. با استفاده از این تکنیک‌ها، ما

توانستیم داده‌های خام FSM را به قالبی تبدیل کنیم که قابل تجزیه و تحلیل و مدل‌سازی باشد.

در این مطالعه، ما یک تحلیل همبستگی برای بررسی روابط بین ویژگی‌های مختلف و تعداد دورهای مورد نیاز برای تأیید هاFSM انجام دادیم. تجزیه و تحلیل ما چندین یافته جالب را نشان داد. ۱.۴ ۱.۴

در ابتدا، ما یک همبستگی مثبت قوی بین ویژگی "All paths" و ستون Round با ضریب همبستگی 0.658 مشاهده کردیم. این نشان می دهد که تعداد مسیرهای ممکن از طریق FSM ها پیش بینی کننده قوی تعداد دورهای تأیید مورد نیاز است. به طور مشابه، همبستگی‌های مثبت متوسطی را بین ویژگی‌های "Num Inputs" (0.375)، "Average Path Length" (0.375) و "Input-to-state Ratio" (0.372) و ستون "Rounds" یافتیم. این ویژگی‌ها جنبه‌های مختلف ساختار و رفتار FSM را نشان می‌دهند، مانند تعداد نمادهای ورودی، طول مسیرهای بین حالت‌ها و نسبت نمادهای ورودی به حالت‌ها.

ما همچنین یک همبستگی منفی متوسط بین ویژگی "Num State" و ستون "Rounds" با ضریب همبستگی (0.467-) مشاهده کردیم. این نشان می دهد که با افزایش تعداد حالت‌ها در ،هاFSM تعداد دورهای تأیید مورد نیاز کاهش می یابد. این یافته در تضاد با همبستگی‌های مثبت مشاهده‌شده برای سایر ویژگی‌ها است و ممکن است مبادله‌ای بین پیچیدگی و قابلیت تأیید در طراحی FSM را نشان دهد.

در نهایت، ما همبستگی‌های مثبت ضعیف تا متوسط را بین ستون 'Rounds' و چندین ویژگی دیگر، از جمله "State Entropy" (0.330) و "Arch Connectivity" (0.173)، و همچنین یک همبستگی مثبت ضعیف با "Self Loops" مشاهده کردیم. از سوی دیگر، ما یک همبستگی منفی متوسط بین ستون "Rounds" و "State Coverage" مشاهده کردیم (0.291-).

به طور کلی، تجزیه و تحلیل همبستگی ما بینش های ارزشمندی را در مورد روابط بین ویژگی ها و تعداد دورهای تأیید مورد نیاز برای FSM ها ارائه کرد. این یافته‌ها می‌توانند برای هدایت انتخاب ویژگی‌های مهم برای مدل‌سازی و پیش‌بینی مورد استفاده قرار گیرند و همچنین می‌توانند به شناسایی مناطق برای بررسی بیشتر و بهبود فرآیندهای طراحی و تأیید FSM کمک کنند.

جدول ۱.۴: ضرایب همبستگی بین ویژگی ها و تعداد دور

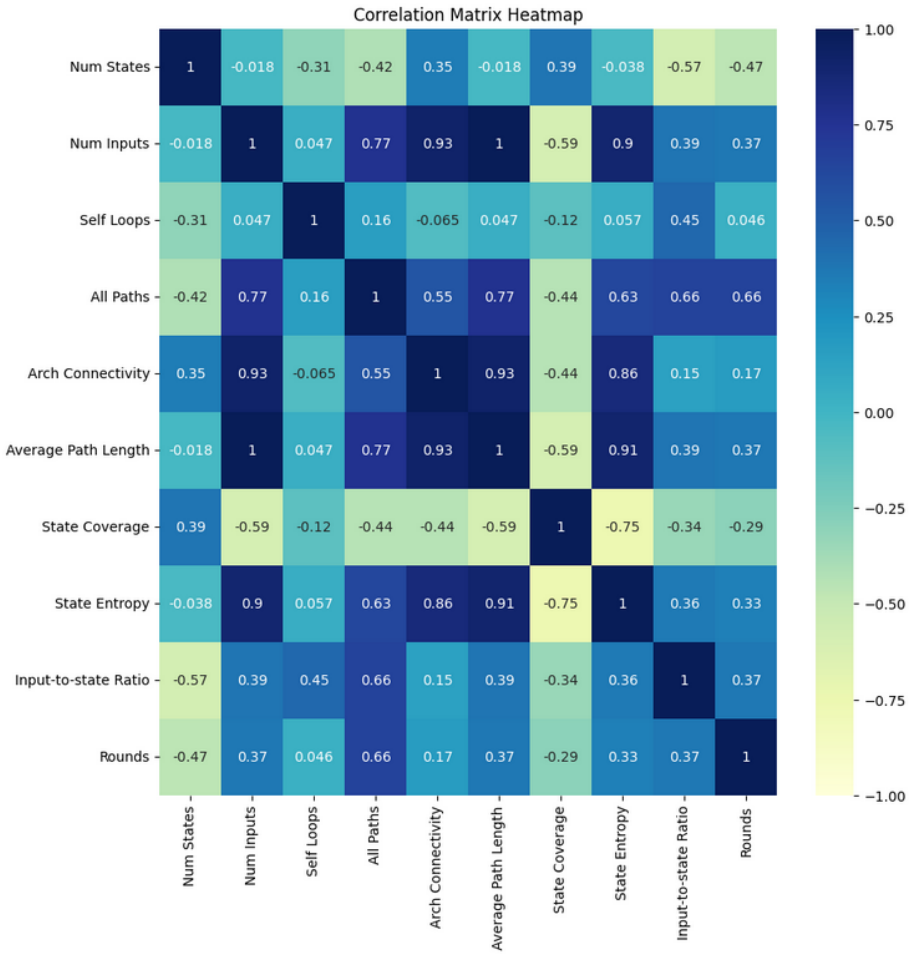
ویژگی	ضریب همبستگی
Rounds	1.000000
All Paths	0.658418
Num Inputs	0.374958
Average Path Length	0.374709
Input-to-state Ratio	0.371812
State Entropy	0.329970
Arch Connectivity	0.172656
Self Loops	0.046104
State Coverage	-0.291266
Num States	-0.467347

۳.۴ نتایج الگوریتم خوشه‌بندی

الگوریتم خوشه‌بندی K-means ابزاری قدرتمند برای گروه‌بندی ویژگی‌های FSM بر اساس شباهت آنها در پیش‌بینی تعداد دورهای تأیید مورد نیاز است. با استفاده از این الگوریتم، ما قادر به شناسایی گروه های متمایز از FSM ها با ویژگی های طراحی و تأیید مختلف بودیم. در این مطالعه، ما از الگوریتم خوشه‌بندی K-means برای گروه‌بندی ویژگی‌های FSM به دو خوشه بر اساس شباهت آنها در پیش‌بینی تعداد دورهای مورد نیاز برای آموزش هاFSM استفاده کردیم. سپس نتایج خوشه‌بندی را با استفاده از طرح ویولن [1] تجسم کردیم ۲.۴ ، که به ما امکان داد توزیع دورهای تأیید را برای هر خوشه مقایسه کنیم. این نوع نمودار به ویژه برای تجسم شکل توزیع ها و شناسایی نقاط پرت بالقوه مفید است.

نتایج ما نشان داد که دو خوشه توزیع‌های متفاوتی از دور تأیید داشتند. خوشه اول که توسط هاییFSM با تعداد کم ورودی‌ها، پیچیدگی مسیر کم و آنتروپی حالت کم مشخص می شد، دارای توزیع کم‌تری از دورهای تأیید بود که از یک تا چهار دور متغیر بود. خوشه دوم، که توسط FSM هایی با تعداد نمادهای ورودی بالا،

[1] Violin plot



شکل ۱.۴: Heatmap ماتریس همبستگی ویژگی های FSM

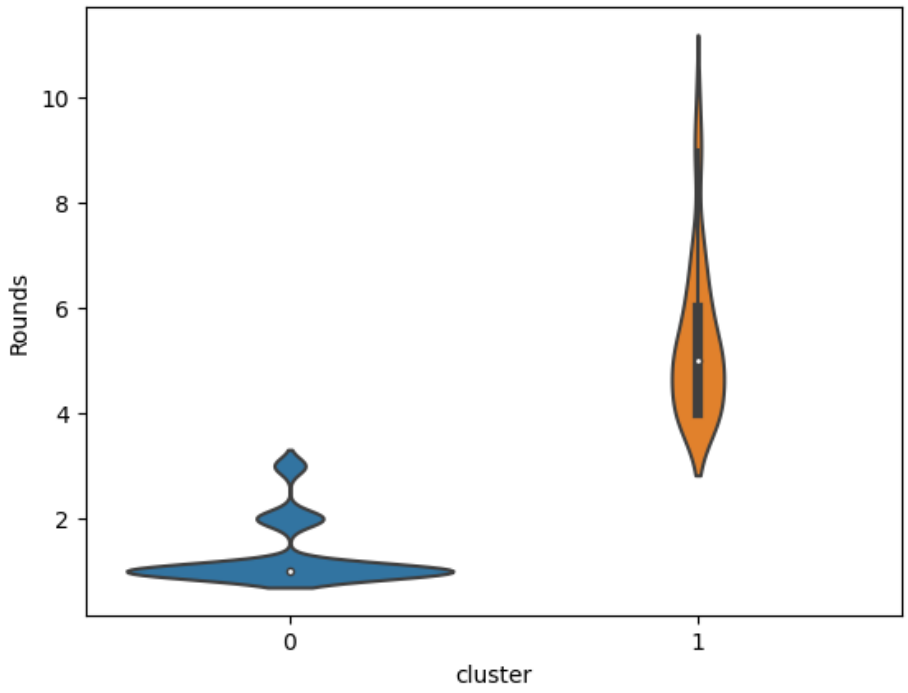
پیچیدگی مسیر بالا و آنتروپی حالت بالا مشخص می شد، دارای توزیع گسترده تری از دورهای تأیید بود که از چهار تا ده دور متغیر بود، با اوج در حدود شش دور.

این نشان می‌دهد که الگوریتم خوشه‌بندی قادر به شناسایی دو گروه متمایز از هاFSM بر اساس پیچیدگی طراحی و تلاش تأیید بود. هاFSM در خوشه اول ممکن است راحت تر تأیید شوند و به دورهای تأیید کمتری نیاز داشته باشند، در حالی که تأیید FSM در خوشه دوم ممکن است دشوارتر باشد و به دورهای تأیید بیشتری نیاز داشته باشد.

به طور کلی، نتایج ما سودمندی بالقوه الگوریتم‌های خوشه‌بندی را در شناسایی گروه‌های FSM با ویژگی‌های طراحی و تأیید مشابه نشان می‌دهد. با در نظر گرفتن این گروه ها در طول فرآیندهای طراحی و تأیید ، FSM طراحان و مهندسان می توانند به طور بالقوه تلاش تأیید را کاهش دهند و کیفیت و قابلیت اطمینان کلی سیستم های خود

را بهبود بخشند.

تحقیقات آینده می‌تواند استفاده از سایر الگوریتم‌های خوشه‌بندی یا تکنیک‌های یادگیری ماشین را برای گروه‌بندی ویژگی‌های FSM و شناسایی الگوها در تلاش‌های راستی‌آزمایی مورد بررسی قرار دهد. علاوه بر این، تحقیقات بیشتر می‌تواند در مورد استراتژی‌های طراحی و راستی‌آزمایی خاص که در کاهش تلاش‌های راستی‌آزمایی برای هاFSM مؤثر است، انجام شود.



شکل ۲.۴: Heatmap ماتریس همبستگی ویژگی های FSM

۴.۴ نتایج الگوریتم رگرسیون جنگل تصادفی

کدی که ارائه کردیم یک مدل رگرسیون جنگل تصادفی را برای پیش‌بینی تعداد دورهای تأیید مورد نیاز برای ماشین‌های حالت محدود (FSM) بر اساس ویژگی‌های ۲.۴ ۳.۴ طراحی آنها آموزش می‌دهد. سپس اهمیت ویژگی ها را برای مدل محاسبه می کند، که نشان دهنده اهمیت نسبی هر ویژگی در پیش بینی خروجی است.

خروجی کد اهمیت ویژگی را برای هر یک از ویژگی های طراحی مورد استفاده در تجزیه و تحلیل نشان می دهد. هرچه مقدار اهمیت بالاتر باشد، ویژگی قدرت پیش بینی بیشتری برای تعداد دورهای تأیید مورد نیاز دارد.

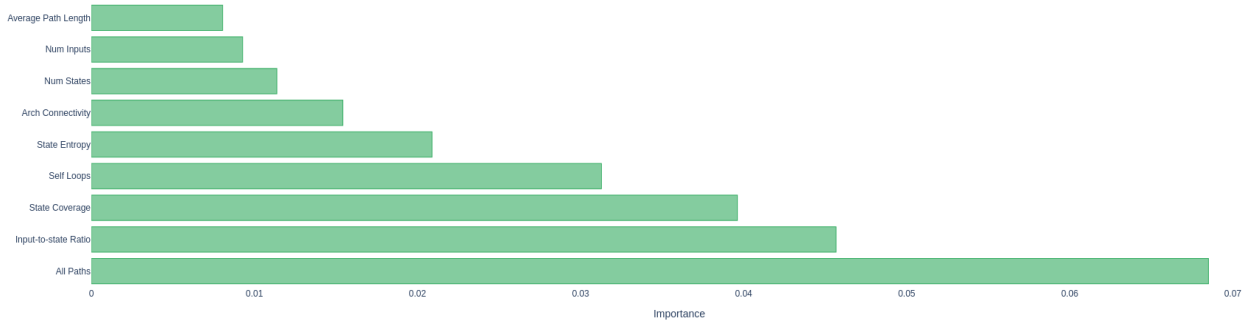
نتایج ما نشان داد که مهم‌ترین ویژگی‌ها برای پیش‌بینی تعداد دورهای تأیید مورد نیاز "All Paths" ، "Input-to-state Ratio" ، "State Coverage" و "Self Loops" بودند. این ویژگی ها به ترتیب دارای ارزش های 0.068 ، 0.046 ، 0.040 و 0.031 بودند. در مقابل، کم‌اهمیت‌ترین ویژگی‌ها عبارت بودند از "Average Path Length" ، "Num Inputs" و "Num States" با مقادیر اهمیت کمتر از 0.01.

این نشان می‌دهد که ویژگی‌های طراحی خاص در پیش‌بینی تعداد دورهای تأیید مورد نیاز برای هاFSM از سایر ویژگی‌ها مهم‌تر هستند. به طور خاص، FSM هایی با تعداد زیادی مسیر ممکن، نسبت ورودی به حالت بالا، پوشش حالت خوب و تعداد حلقه های self-loop زیاد ممکن است نسبت به FSM های بدون این ویژگی ها به دور تأیید بیشتری نیاز داشته باشند.

به طور کلی، نتایج ما سودمندی بالقوه الگوریتم‌های یادگیری ماشین را در پیش‌بینی تلاش تأیید مورد نیاز برای هاFSM بر اساس ویژگی‌های طراحی آنها نشان می‌دهد. با در نظر گرفتن این ویژگی ها در طول طراحی ،FSM طراحان و مهندسان می توانند به طور بالقوه تلاش تأیید را کاهش دهند و کیفیت و قابلیت اطمینان کلی سیستم های خود را بهبود بخشند.

جدول ۲.۴: اهمیت ویژگی برای پیش‌بینی تلاش راستی‌آزمایی FSM

ویژگی	اهمیت
Num States	0.0114
Num Inputs	0.0093
Self Loops	0.0313
All Paths	0.0685
Arch Connectivity	0.0154
Average Path Length	0.0080
State Coverage	0.0396
State Entropy	0.0209
Input-to-state Ratio	0.0457



شکل ۳.۴: اهمیت ویژگی برای پیش‌بینی تلاش راستی‌آزمایی FSM

فصل ۵

بحث و نتیجه‌گیری

۱.۵ مقدمه

در این مقاله، ما استفاده از تکنیک‌های یادگیری ماشین را برای پیش‌بینی تلاش مورد نیاز برای ماشین‌های حالت محدود (FSM) بررسی کردیم. ما با بحث در مورد اهمیت تأیید FSM و چالش‌های مرتبط با آن، از جمله افزایش پیچیدگی سیستم‌ها، شروع کردیم. سپس مفهوم انتخاب ویژگی را معرفی کردیم و ویژگی‌های طراحی مختلفی را مورد بحث قرار دادیم که می‌توانند بر تلاش تأیید برای هاFSM تأثیر بگذارند.

در مرحله بعد، ما یک رویکرد مبتنی بر یادگیری ماشین برای پیش‌بینی تعداد دورهای تأیید مورد نیاز برای هاFSM با استفاده از رگرسیون جنگل تصادفی ارائه کردیم. ما آموزش و ارزیابی مدل رگرسیون را مورد بحث قرار دادیم و اهمیت ویژگی‌ها را برای شناسایی حیاتی‌ترین ویژگی‌های طراحی برای تأیید FSM تجزیه و تحلیل کردیم.

نتایج ما نشان داد که رویکرد پیشنهادی به دقت بالایی در پیش‌بینی تلاش تأیید برای هاFSM دست یافت و ویژگی‌های طراحی شناسایی‌شده را می‌توان برای بهینه‌سازی فرآیند تأیید و کاهش زمان و هزینه مورد نیاز برای تأیید FSM استفاده کرد.

به طور کلی، مطالعه ما پتانسیل تکنیک‌های یادگیری ماشین را برای بهبود کارایی و اثربخشی تأیید FSM برجسته می‌کند و راه‌های جدیدی را برای تحقیقات آینده در این زمینه باز می‌کند.

۲.۵ محتوا

۱.۲.۵ جمع‌بندی

مطالعه ما کارایی تکنیک‌های یادگیری ماشین را برای پیش‌بینی تلاش مورد نیاز برای ماشین‌های حالت محدود (FSM) نشان می‌دهد. با تجزیه و تحلیل ویژگی‌های طراحی که بر تلاش راستی‌آزمایی تأثیر می‌گذارند و آموزش یک مدل رگرسیون جنگل تصادفی، توانستیم تعداد دورهای تأیید مورد نیاز برای هاFSM را با دقت بالایی پیش‌بینی کنیم.

یافته‌های ما پتانسیل یادگیری ماشین را برای بهبود کارایی و اثربخشی تأیید ،FSM که یک گام اساسی در طراحی و توسعه سیستم‌ها است، برجسته می‌کند. با کاهش تلاش تأیید مورد نیاز برای ،هاFSM می‌توانیم فرآیند تأیید را بهینه کنیم، در زمان و منابع صرفه‌جویی کنیم و کیفیت و قابلیت اطمینان کلی سیستم‌های دیجیتال را بهبود بخشیم.

۲.۲.۵ نوآوری

مطالعه ما چندین رویکرد نوآورانه را برای پیش‌بینی تلاش تأیید مورد نیاز برای ماشین‌های حالت محدود (FSM) معرفی می‌کند.

۱. در مرحله اول، ما از تکنیک‌های یادگیری ماشین برای خودکارسازی فرآیند پیش‌بینی استفاده کردیم، که می‌تواند زمان و تلاش مورد نیاز برای تأیید FSM را به میزان قابل توجهی کاهش دهد.

۲. ثانیا، ما از تکنیک‌های انتخاب ویژگی برای شناسایی ویژگی‌های طراحی حیاتی که بر تلاش تأیید تأثیر می‌گذارند، استفاده کردیم، که می‌تواند به بهینه‌سازی فرآیند تأیید و بهبود کارایی تأیید FSM کمک کند.

۳. ثالثا، ما از رگرسیون جنگل تصادفی برای پیش‌بینی دقیق تعداد دورهای تأیید مورد نیاز برای هاFSM استفاده کردیم. این رویکرد می‌تواند تخمین دقیق‌تری از تلاش تأیید مورد نیاز برای یک طراحی FSM ارائه دهد، که می‌تواند به کاهش خطر خطاها یا نقص‌های طراحی و بهبود کیفیت کلی و قابلیت اطمینان سیستم‌های دیجیتال کمک کند.

به طور کلی، مطالعه ما نشان دهنده پیشرفت قابل توجهی در زمینه تأیید FSM و طراحی سیستم است. با استفاده از تکنیک‌های یادگیری ماشین و انتخاب ویژگی، می‌توانیم کارایی و اثربخشی تأیید FSM را بهبود بخشیم، زمان و هزینه مورد نیاز برای تأیید FSM را کاهش دهیم و راه را برای تحقیقات و نوآوری‌های آینده در این زمینه هموار کنیم.

۳.۲.۵ پیشنهادها

در حالی که مطالعه ما یک رویکرد امیدوارکننده برای پیش‌بینی تلاش تأیید مورد نیاز برای ماشین‌های حالت محدود (FSM) ارائه می‌کند، حوزه‌های مختلفی وجود دارد که تحقیقات آینده می‌تواند بر اساس کار ما باشد. در مرحله اول، مطالعه ما بر پیش‌بینی تلاش تأیید برای هاFSM با استفاده از یک مدل رگرسیون جنگل تصادفی متمرکز بود. مطالعات آینده می تواند استفاده از سایر الگوریتم های یادگیری ماشین یا رویکردهای ترکیبی را برای بهبود دقت و کارایی تأیید FSM مورد بررسی قرار دهد.

همچنین، مطالعه ما از مجموعه محدودی از ویژگی‌های طراحی برای پیش‌بینی تلاش تأیید برای هاFSM استفاده کرد. مطالعات آتی می‌تواند استفاده از ویژگی‌های طراحی اضافی، مانند ویژگی‌های ساختاری و رفتاری هاFSM یا در دسترس بودن مجموعه‌های آزمایشی را برای بهبود دقت پیش‌بینی تلاش راستی‌آزمایی مورد بررسی قرار دهد.

مطالعه ما بر پیش‌بینی تلاش تأیید مورد نیاز برای هاFSM با تعداد محدودی حالت و ورودی متمرکز بود. مطالعات آینده می‌تواند مقیاس‌پذیری رویکرد ما را برای طرح‌های بزرگ‌تر و پیچیده‌تر ،FSM از جمله مواردی که در سیستم‌های حیاتی ایمنی، مانند سیستم‌های کنترل هواپیما یا دستگاه‌های پزشکی استفاده می‌شوند، بررسی کند.

۴.۲.۵ محدودیت‌ها

مطالعه شما دارای چندین محدودیت است که باید در تفسیر نتایج ما در نظر گرفته شود. اولاً، مطالعه ما تنها بر پیش‌بینی تلاش تأیید مورد نیاز برای ماشین‌های حالت محدود (FSM) با استفاده از مجموعه محدودی از

ویژگی‌های طراحی متمرکز بود. عوامل دیگری مانند پیچیدگی منطق کنترل [1] سیستم یا در دسترس بودن ابزارهای تأیید نیز ممکن است بر تلاش تأیید مورد نیاز برای FSM تأثیر بگذارد.

ثانیاً، مطالعه ما بر مجموعه داده‌ای از طرح‌های FSM تکیه داشت که ممکن است نماینده همه طرح‌های ممکن FSM یا برنامه‌های کاربردی دنیای واقعی نباشد. مطالعات آینده می‌تواند تعمیم‌پذیری رویکرد ما به انواع مختلف طرح‌ها و برنامه‌های FSM را بررسی کند.

ثالثاً، مطالعه ما تأثیر پیاده‌سازی سخت‌افزار بر تلاش تأیید مورد نیاز برای هاFSM را در نظر نگرفت. تلاش تأیید برای طراحی FSM ممکن است تحت تأثیر پلت فرم سخت افزاری مورد استفاده برای پیاده سازی قرار گیرد، و تحقیقات آینده می تواند تأثیر پیاده سازی سخت افزار را بر تلاش تأیید مورد نیاز برای FSM ها بررسی کند.

[1] Control logic

کتاب‌نامه

[1] Angluin, Dana. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[2] Björklund, Johanna, Fernau, Henning, and Kasprzik, Anna. Polynomial inference of universal automata from membership and equivalence queries. *Information and Computation*, 246:3–19, 2016. 7th International Conference on Language and Automata Theory and Applications (LATA 2013).

[3] Damasceno, Carlos Diego N, Mousavi, Mohammad Reza, and da Silva Simão, Adenilso. Learning to reuse: Adaptive model learning for evolving systems. In *Integrated Formal Methods: 15th International Conference, IFM 2019, Bergen, Norway, December 2–6, 2019, Proceedings*, pages 138–156. Springer, 2019.

[4] Lity, Sascha, Lachmann, Remo, Lochau, Malte, and Schaefer, Ina. Delta-oriented software product line test models-the body comfort system case study. *Technical report, TU Braunschweig*, 2013.

[5] Tavassoli, Shaghayegh, Damasceno, Carlos Diego N, Khosravi, Ramtin, and Mousavi, Mohammad Reza. Adaptive behavioral model learning for software product lines. In *Proceedings of the 26th ACM International Systems and Software Product Line Conference-Volume A*, pages 142–153, 2022.

[6] Tavassoli, Shaghayegh, Damasceno, Carlos Diego N, Mousavi, Mohammad Reza, and Khosravi, Ramtin. A benchmark for active learning of variability-intensive systems. In *Proceedings of the 26th ACM International Systems and Software Product Line Conference-Volume A*, pages 245–249, 2022.

[7] Tsarev, Fedor and Egorov, Kirill. Finite state machine induction using genetic algorithm based on testing and model checking. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '11, page 759–762, New York, NY, USA, 2011. Association for Computing Machinery.