DOR: 20.1001.1.2538161.1403.9.4.5.0
DOI: https://doi.org/10.22034/csj.2025.214936
زمستان ۱۴۰۳، دوره ۹، شماره ۴، پیاپی ۳۵

دریافت مقاله: ۱۴۰۳/۰۹/۲۰
پذیرش مقاله: ۱۴۰۳/۱۰/۲۹

نوع مقاله: پژوهشی

کاربرد هوش مصنوعی قابل توضیح برای تحلیل نیازمندی‌های نرم‌افزار

بهناز جاماسب
دانشجوی دکتری، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی شیراز
پست الکترونیکی: b.jamasb@sutech.ac.ir

سید رئوف خیامی*
دانشیار، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی شیراز، شیراز، ایران
پست الکترونیکی: khayami@sutech.ac.ir

رضا اکبری
دانشیار، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی شیراز، شیراز، ایران
پست الکترونیکی: akbari@sutech.ac.ir

چکیده

الگوریتم‌های یادگیری ماشین در مسایل مختلف مانند طبقه‌بندی نیازمندی‌ها استفاده شده‌اند. با وجود این که این الگوریتم‌ها عملکرد خوبی دارند، اما نمی‌توانند نحوه انجام تصمیم‌گیریشان را توضیح دهند. توضیح نحوه تصمیم‌گیری کمک بزرگی برای متخصصین این حوزه خواهد بود. هوش مصنوعی قابل توضیح (XAI) به دنبال توضیح پیش‌بینی‌های الگوریتم‌های یادگیری ماشین است. با توجه به کاربرد موفق هوش مصنوعی قابل توضیح در حوزه پردازش زبان طبیعی و عدم وجود مطالعات کافی در زمینه به‌کارگیری آنها در تحلیل نیازمندی‌های نرم‌افزار، در این مقاله، کاربرد XAI برای طبقه‌بندی نیازمندی‌های نرم‌افزار بررسی شده است. روش پیشنهادی روی مجموعه داده نیازمندی‌های نرم‌افزار PROMISE پیاده‌سازی شده است. الگوریتم ماشین بردار پشتیبان (SVM) به‌عنوان الگوریتم طبقه‌بندی استفاده شده و ابزارLIME به تفسیر دسته‌بندی نیازمندی‌های تولید شده توسط طبقه‌بند می‌پردازد. نتایج نشان می‌دهد که XAI می‌تواند به تحلیل‌گر کمک کند تا بهتر درک کند که چرا یک نیازمندی خاص به‌عنوان کارکردی، غیرکارکردی یا زیرشاخه‌ای از غیرکارکردی طبقه‌بندی شده است. در طی این فرآیند، کلمات کلیدی مهم برای چنین تصمیماتی شناسایی شده و به‌طور دقیق تحلیل می‌شوند.

کلمات کلیدی: هوش مصنوعی قابل توضیح، تحلیل نیازمندی نرم‌افزار، طبقه‌بندی نیازمندی‌ها

۱. مقدمه

مهندسی نیازمندی‌ها[1] یکی از مراحل اصلی در چرخه حیات توسعه نرم‌افزار است و شامل فعالیت‌های مختلفی است که توسط تحلیل‌گران نرم‌افزار انجام می‌شود. استخراج نیازمندی‌ها، تحلیل، اعتبارسنجی و مدیریت معمولاً به عنوان فعالیت‌های عمومی در مهندسی نیازمندی‌ها شناخته می‌شوند [۱]. فعالیت استخراج نیازمندی‌ها به شناسایی

1-Requirements Engineering

* نویسندهٔ مسئول

خدمـــاتی که نرم‌افزار باید ارایه دهد، همراه با محدودیت‌ها و معیارهـــای کیفیتی که باید توســـط نرم‌افـــزار برآورده شود، می‌پردازد. این فعالیت شامل مراحل مختلفی از جمله کشـــف نیازمندی‌ها، طبقه‌بنـــدی نیازمندی‌ها، اولویت‌بندی نیازمندی‌ها و مشخصات نیازمندی‌ها است.

فعالیت اســـتخراج نیازمندی‌ها تأثیر مهـــمی بر کیفیت نهـــایی نرم‌افزار دارد. هرگونه نقـــص در این فعالیت مانند نیازمندی‌های از دست رفته، نیازمندی‌های غیر قابل پیگیری، نیازمندی‌های متناقض، ابهام در نیازمندی‌ها و غیره، شانس توســـعه نرم‌افزار با کیفیت بالا را کاهـــش می‌دهد. کیفیت نرم‌افزار به نیازمندی‌های صحیح وابســـته اســـت و چون اســـتخراج نیازمندی‌ها یک کار زمان‌بر اســـت، استفاده از روش‌ها و راه‌حل‌های خودکار جهت استخراج نیازمندی‌ها، پیشنهاد می‌شود. به‌عنوان مثال، روش‌های یادگیری ماشین ماننـــد طبقه‌بندی‌کنندگان متنی می‌توانند برای دســـته‌بندی نیازمندی‌های کارکـــردی[۲] و نیازمندی‌های غیرکارکردی[۳] مورد استفاده قرار گیرند.

معمولاً نیازمندی‌های نرم‌افزار به دو دســـته کارکردی و غیرکارکـــردی دســـته‌بندی می‌شـــوند. نیازمندی‌هـــای کارکـــردی مســـئولیت‌ها یا وظایـــف نرم‌افزار را نشـــان می‌دهند و نیازمندی‌هـــای غیرکارکردی نمایانگر کیفیت یا محدودیت‌هایی هســـتند که نرم‌افزار در حین انجام وظایف خود بایـــد آن‌ها را بـــرآورده کند. بیشـــتر نیازمندی‌های غیرکارکـــردی معیارهای کیفیـــت نرم‌افـــزار مانند امنیت، عملکرد، قابلیت اطمینان، قابلیت اســـتفاده و غیره را نشان می‌دهند [۱۸].

در سال‌های اخیر، تأثیر هوش مصنوعی[۴] بر حوزه‌های مختلف افزایش یافته اســـت. بســـیاری از راه‌حل‌های هوش مصنوعی توسط پژوهشگران ارایه شده‌اند تا مشکلات پیچیده را حل کنند و به افراد و متخصصان حوزه‌های مختلف کمک کنند تا کارهای خود را بهتر انجام دهند. یادگیری ماشین یک زیرمجموعه از هوش مصنوعی اســـت که شامل دسته‌های مختلف روش‌های نظارت شـــده[۵]، نظارت نشده[۶] و تقویتی[۷] است.

هوش مصنوعی قابل توضیح (XAI)[۸]، یک زیرمجموعه از هوش مصنوعی اســـت کـــه کمک می‌کند سیســـتم‌های هوش مصنوعی شفاف و قابل درک برای انسان‌ها توسعه داده شـــوند. هوش مصنوعی قابل توضیح نقش مهمی در حوزه‌های مختلف دارد و به انسان‌ها این امکان را می‌دهد تا اســـتدلال پشت نتایج تولید شده توسط هوش مصنوعی را درک کنند. به این ترتیب، XAI اشـــکال‌زدایی و عیب‌یابی را تســـهیل می‌کند، انطباق با قوانیـــن را تضمین می‌کند و در نهایت باعث توســـعه سیستم‌های هوش مصنوعی قابل اعتماد و قابل اطمینان می‌شود.

روش‌های یادگیری نظارت‌شـــده که شـــامل طبقه‌بندی و رگرسیون هســـتند، به‌طـــور موفقیت‌آمیـــزی برای حل مشـــکلات مهندسی نرم‌افزار به‌کار رفته‌اند. با این حال، در بیشتر موارد این روش‌ها به‌عنوان یک جعبه سیاه استفاده می‌شـــوند و راه‌حل‌های آن‌ها قابل تفسیر نیستند. به دلیل اهمیت تحلیـــل نیازمندی‌ها، قابلیـــت اطمینان به مدل جعبه سیاه بسیار مهم اســـت. XAI می‌تواند کمـــک کند تا درجه اطمینان روش‌های نظارت‌شده در طبقه‌بندی نیازمندی‌ها را تعیین کنیـــم. با وجود به‌کارگیری موفق XAI در حوزه‌های مختلـــف پردازش زبان طبیعی و تحلیـــل متن، اما تحقیقات بسیار محـــدودی در زمینه اســـتفاده از XAI جهت تفسیر نتایـــج الگوریتم‌های یادگیری ماشین کـــه برای طبقه‌بندی نیازمندی‌های نرم‌افزار اســـتفاده می‌شـــوند، گزارش شده اســـت. با توجه به این که پردازش نیازمندی‌های نرم‌افزار را می‌توان زیرمجموعه‌ای از مســـایل مورد بحث در حوزه پردازش زبان طبیعی در نظر گرفت، اما ویژگی‌های منحصر بفردی در این مســـئله وجود دارد که در ســـایر حوزه‌های پردازش زبان طبیعی وجود نـــدارد. برای مثال کلماتی که اولویـــت نیازمندی‌هـــا را تعیین می‌کنند، یـــا واژه‌هایی که

2-Functional Requirements
3-Non-functional Requirements
4-Artificial Intelligence

5-Supervised Learning
6-Unsupervised Learning
7-Reinforcement
8-Explainable Artificial Intelligence

نشـــان‌دهنده نوع نیازمندی‌های عملکردی و غیرعملکردی هستند نشـــان‌دهنده تفاوت‌هایی است که به‌طور خاص در پردازش نیازمندی‌ها دارای اهمیت هستند. بنابراین می‌توان ایـــن مورد را به‌عنـــوان یکی از موضوعات بـــاز در زمینه پـــردازش نیازمندی‌های نرم‌افزار کـــه به زبان طبیعی بیان می‌شـــوند در نظر گرفت. این مســـئله نویسندگان مقاله را بر این داشـــته است تا یک مدل توضیح‌دهنده شناخته شده برای توضیح الگوریتم‌هـــای طبقه‌بندی نیازمندی نرم‌افزار ارایه دهند.

در این پژوهش روشی برای توضیح نتایج تولید شـــده توســـط یک الگوریتم طبقه‌بندی نیازمنـــدی ارایه می‌گردد. الگوریتم‌هـــای طبقه‌بندی مختلفی وجود دارند که بخشی از آن‌ها مانند درخت تصمیم[9] و رگرسیون خطی[10] قابل توضیح هســـتند، در حالی که برخی دیگر مانند شـــبکه‌های عصبی عمیق[11]، ماشین‌های بردار پشـــتیبان[12] و جنگل تصادفی[13] به دلیـــل پیچیدگی ذاتی یـــا عدم شـــفافیت، قابل توضیح نیســـتند. معمولاً ایـــن طبقه‌بندهای پیچیده در مقایســـه با مدل‌های ســـاده و قابل توضیح، نتایج بهتری ارایه می‌دهند و می‌توانند الگوهای پیچیده و تعاملات آن‌ها را درک کنند. بنابراین، اســـتفاده از مدل‌های پیچیده به دلیل عملکردشان خوب است و XAI می‌تواند کمبود ذاتی تفسیرپذیری آن‌ها را برطرف کند.

در اینجا از طبقه‌بند SVM اســـتفاده شـــده است [۲]. با ایـــن حال، امـــکان جایگزینی SVM با ســـایر الگوریتم‌های طبقه‌بنـــدی پیچیده وجود دارد. هم‌چنیـــن از توضیح‌دهنده LIME بـــه همراه دانش متخصصان این حوزه برای تفسیر دسته‌بندی نیازمندی‌هایی که توســـط الگوریتم طبقه‌بندی تولید شده، اســـتفاده شده است. برای این منظور، ما روی کلمات کلیدی در نیازمندی‌ها که از راه‌حل طبقه‌بند موردنظر حمایـــت می‌کنند یا با آن مخالفت دارند، تمرکز می‌کنیم. در نهایـــت به متخصصان حوزه کمک می‌شـــود تا بدانند چرا یک نیازمندی به‌عنوان کارکردی یا غیرکارکردی طبقه‌بندی شده است. نوآوری‌های اصلی این مقاله عبارتند از:

- بـــرای اولین بـــار از XAI بـــرای توضیـــح طبقه‌بندهای نیازمندی‌های نرم‌افزار استفاده شده است.
- یـــک چارچـــوب بـــرای توضیـــح الگوریتـــم طبقه‌بندی نیازمندی‌های نرم‌افزار پیشنهاد شده است.
- یک مدل توضیح‌دهنده پیشـــرفته بـــرای توضیح کیفیت طبقه‌بندی‌کننده نیازمندی‌های نرم‌افزار استفاده شده است.

ساختار مقاله به این صورت است: بخش ۲ به کارهای پیشین می‌پـــردازد. معرفی XAI و الگوریتم LIME در بخش ۳ ارایه می‌شـــود. بخش ٤ روش پیشنهادی را ارایه می‌کند. بخش ٥ به کاربرد مدل و نتایج تجربی اختصاص دارد. در نهایت، در بخش ٦ نتیجه‌گیری این کار ارایه می‌شود.

۲. مطالعات پیشین

مســـئلهٔ طبقه‌بندی نیازمندی‌های نرم‌افـــزار را می‌توان به‌عنوان یکی از مســـایل موجـــود در حوزه پردازش زبان طبیـــعی در نظـــر گرفت. زیـــرا نیازمندی‌هـــای نرم‌افزاری معمولاً بـــه زبان طبیعی بیان می‌شـــوند، چـــه به صورت مستندات رسمی (مانند مشخصات نیازمندی‌های نرم‌افزار) و چه به شـــکل‌های غیررسمی (مانند داستان‌های کاربر یا درخواست‌های ویژگی). این وظیفه شامل پردازش و درک این متون زبان طبیعی برای دسته‌بندی آن‌ها به گروه‌های معناداری مانند نیازمندی‌هـــای عملکردی و نیازمندی‌های غیرعملکردی اســـت. لذا برخی از مطالعات انجام شده در ایـــن حوزه با تمرکز بر هوش مصنوعی قابل توضیح ارایه می‌شود.

اخیـــراً، کاربردهای هوش مصنـــوعی قابل توضیح در زمینه پردازش داده‌های متنی و پردازش زبان طبیعی مورد مطالعه قرار گرفته اســـت. برای مثال، دیـــوالی و همکاران مروری را بر روی روش‌های تحلیل احساسات با تمرکز بر استفاده از هوش مصنوعی قابل توضیح ارایه داده‌اند[۱۹]. در مطالعه دیگری، ماتئـــوس کاربردهای هوش مصنوعی

9-Decision Tree
10-Linear Regression
11-Deep Neural Networks
12-Support Vector Machines
13-Random Forest

قابل توضیح را در حوزه‌های پردازش زبان طبیعی، زیست پزشکی و تشخیص بدافزار بررسی کرده است [۲۰]. ارزیابی وی بر روی پردازش زبان طبیعی برای طبقه‌بندی داده‌های توییت انجام می‌شود، طبقه‌بندی سیگنال‌های زیست‌پزشکی برای دسته‌بندی سرطان و طبقه‌بندی بدافزار رایانه شخصی است. زهور و زکریا از رهیافتی مبتنی بر هوش مصنوعی قابل توضیح برای دسته‌بندی نظرات کاربران در مورد برنامه کاربردی موبایل استفاده کرده‌اند [۲۱]. ژانگ و آوگاساوارا از یک روش مبتنی بر هوش مصنوعی قابل توضیح برای دسته‌بندی متون پزشکی استفاده کرده‌اند. آنها نشان داده‌اند الگوریتم موردنظر از چه کلماتی برای دسته‌بندی استفاده می‌کند[۲۲]. در پژوهش دیگری، از هوش مصنوعی قابل توضیح در پردازش زبان طبیعی استفاده شده است. نویسندگان کارایی دو روش توضیح مدل‌محور پر کاربرد را بر روی مجموعه داده جملات محاوره‌ای اعمال کرده‌اند و نتیجه آنها را گزارش داده‌اند[۲۳]. این مطالعات نشان می‌دهند هوش مصنوعی قابل توضیح به‌طور موفق برای حل مسایل موجود در حوزه پردازش زبان طبیعی قابل استفاده هستند و لذا می‌توان از آنها برای تحلیل نیازمندی‌های نرم‌افزار به همراه الگوریتم‌های طبقه‌بندی یادگیری ماشین استفاده کرد.

در سال‌های اخیر از الگوریتم‌های مختلف نظارت‌شده توسط پژوهشگران برای طبقه‌بندی نیازمندی‌ها استفاده شده است. در ادامه به کارهای انجام شده در زمینه طبقه‌بندی نیازمندی‌ها و کاربرد هوش مصنوعی قابل توضیح در حوزه مهندسی نیازمندی‌ها پرداخته می‌شود. بر اساس تحقیقات و اطلاعات نویسندگان مقاله حاضر، مطالعات بسیار محدودی در زمینه کاربرد هوش مصنوعی قابل توضیح برای تحلیل نیازمندی‌های نرم‌افزار ارایه شده است. در ابتدا برخی از مطالعات مربوط به الگوریتم‌های طبقه‌بندی یادگیری ماشین جهت طبقه‌بندی نیازمندی‌ها ارایه می‌شود و سپس مطالعات مرتبط به هوش مصنوعی قابل توضیح در ارتباط با مهندسی نیازمندی‌ها مطرح می‌گردد.

بین‌خونین و ژائو مروری نظام‌مند بر روش‌های یادگیری ماشین برای شناسایی و طبقه‌بندی نیازمندی‌های غیرکارکردی ارایه کردند[۳]. مطالعات آن‌ها نشان داد که ۱۶ روش مختلف یادگیری ماشین توسط پژوهشگران استفاده شده و الگوریتم‌های نظارت‌شده محبوب‌ترین آن‌ها هستند. آن‌ها ذکر کردند که در حالی که روش‌های یادگیری ماشین در تحلیل نیازمندی‌ها موفق بوده‌اند، با چالش‌های باز در این حوزه مواجه هستیم که می‌توان آن‌ها را با همکاری بین پژوهشگران مهندسی نرم‌افزار و یادگیری ماشین حل کرد

خیاشی و همکاران از تکنیک‌های مختلف یادگیری عمیق برای طبقه‌بندی نیازمندی‌های نرم‌افزار به عنوان کارکردی یا غیرکارکردی استفاده کردند [٤]. آن‌ها پنج الگوریتم یادگیری عمیق همراه با دو روش رای‌گیری[14] را روی مجموعه داده PURE اعمال کردند. نتایج نشان داد که اعمال روش‌های رای‌گیری بر روی الگوریتم‌های یادگیری عمیق نتایج بهتری تولید می‌کند. در پژوهشی دیگر، کانیدو و مندس به بررسی ترکیب سه روش استخراج ویژگی به نام‌های بسته کلمات(BoW)[15] ، فراوانی اصطلاح- معکوس فراوانی متن(TF-IDF)[16] و کای‌دو (CHI^2)[17] با چهار الگوریتم طبقه‌بندی، ماشین بردار پشتیبان (SVM)[18]، بیز ساده چندجمله‌ای[19]، نزدیک‌ترین همسایه[20] و رگرسیون لجستیک[21] پرداختند [٥]. آن‌ها دریافتند که ترکیب TF-IDF و رگرسیون لجستیک بهترین عملکرد را در طبقه‌بندی دوتایی دارد.

طبقه‌بندی نیازمندی‌های کارکردی توسط رحیمی و همکاران بررسی شد [٦]. آن‌ها یک روش تجمعی با ترکیب پنج الگوریتم طبقه‌بندی پیشنهاد کردند. در کار دیگری، آن‌ها الگوریتم‌های مختلف یادگیری عمیق را برای طبقه‌بندی نیازمندی‌ها در روش‌های یک‌مرحله‌ای و دومرحله‌ای اعمال کردند. ابتدا نیازمندی‌ها را به‌عنوان کارکردی و غیرکارکردی طبقه‌بندی کردند. سپس

14-Voting Method
15-Bag of Words
16-Term Frequency–Inverse Document Frequency
17-Chi-Squared
18-Support Vector Machine
19-Multinomial Naive Bayes
20-k-Nearest Neighbors
21-Logistic Regression

نیازمندی‌هـــای غیرکارکـــردی را زیرشـــاخه‌بندی کردند. همچنین قوبـــا و همکاران به بررسی عملکرد ترکیب BoW بـــا SVM و KNN برای طبقه‌بنـــدی نیازمندی‌های نرم‌افزار پرداختند [۷]. مشـــخص شد که ترکیب BoW و SVM نتایج بهتری تولید می‌کند.

مطالعات قبلی نشان دادند که روش‌های یادگیری ماشین در طبقه‌بنـــدی نیازمندی‌ها عملکرد خـــوبی دارند. با وجود کاربرد موفقیت‌آمیز یادگیری ماشین، این روش‌ها به‌عنوان راه‌حل‌هـــای جعبه سیاه اســـتفاده می‌شـــوند. این روش‌ها به دلیل عدم شـــفافیت برای کاربران، جعبـــه سیاه نامیده می‌شـــوند. فرض کنید که یک روش یادگیری نظارت‌شـــده پیچیده داشته باشیم. به دلیل پیچیدگی آن، نمی‌توانیم درک کنیم که الگوریتم چگونه نتیجه خاصی را تولید می‌کند. در برخی موارد، یک متخصص حوزه نیاز دارد بداند الگوریتم چگونه به چنین نتیجه‌ای رسیده است تا بتواند به آن اعتماد کند.

در اینجا، XAI وارد عمل می‌شود. XAI روشی است که از روش‌ها و تکنیک‌های مختلف برای توضیح مدل‌های هوش مصنوعی استفاده می‌کند. در سال‌های اخیر، روش‌های XAI مختلفی پیشنهاد شده‌اند. توضیحات مدل‌های محلی و مستقل از الگوریتم (LIME)[۲۲] و توضیحات افزودنی شاپلی (SHAP)[۲۳] از جمله متداول‌ترین توضیح‌دهنده‌ها هستند [۸]. هر دوی آن‌ها به‌عنوان مدل‌های جانشین شـــناخته می‌شـــوند و از مدل‌های یادگیری ماشین جعبه سیاه استفاده می‌کنند.

LIME توســـط ریبیرو و همکاران ارایه شـــد [۹]. هدف LIME توضیح نتایج تولیدشـــده توسط هر مدل طبقه‌بندی یا رگرسیون اســـت. LIME یـــک پیش‌بینی را با تقریب زدن آن به‌صورت محـــلی توضیح می‌دهد. البته الگوریتم LIME که به‌طور گسترده‌ای برای توضیح پیش‌بینی‌های مدل‌های یادگیری ماشین استفاده می‌شود، با چندین چالش روبه‌رو اســـت که روی قابلیت‌های آن تاثیرگذار است. کارایی این الگوریتم به دلیل تصادفی بودن در نمونه‌برداری تغییرات، تحت تاثیر قرار می‌گیرد و فاقد بینش‌های سراسری است، زیـــرا تنها بر روی رفتارهای محـــلی تمرکز می‌کند. کیفیت توضیحات ارایه شـــده توســـط این الگوریتم بـــه راهبرد تغییرات بستگی دارد، بنابراین ممکن است اثرات منفی روی توضیحات ارایه شده برای داده‌های متنی باشد. استفاده از مدل‌های خطی به‌عنوان مدل‌های جایگزین، می‌تواند مرزهای تصمیم‌گیـــری پیچیده را ساده‌ســـازی کنـــد و تعادل بین تفسیرپذیری و دقت معمولاً منجر به توضیحاتی می‌شـــود که یا بیش از حد ساده یا بسیار پیچیده هستند. LIME از نظر محاسباتی پرهزینه است، به مقیاس‌بندی ویژگی‌ها حساس اســـت و در برابر نمونه‌های خصمانه آسیب‌پذیر است، که می‌تواند منجر به توضیحات گمراه‌کننده شـــود. علاوه بر این، هیچ معیاری برای کمی‌ســـازی اعتبار توضیحات آن وجود ندارد. رسیدگی به این مشکلات از طریق بهبود ثبات، روش‌هـــای قوی‌تر برای ایجاد تغییرات، مدل‌های جایگزین پیشرفته و معیارهای بهتر می‌تواند قابلیت اطمینان و مفید بودن LIME را افزایش دهد.

لونبـــرگ و لی الگوریتـــم SHAP را ارایـــه کردند [۱۰]. مشـــابه LIME، الگوریتم SHAP می‌توانـــد توضیح دهد که هـــر ویژگی یک الگوریتم طبقه‌بندی یا رگرسیون چگونه به یـــک پیش‌بینی کمک می‌کند. SHAP یک چارچوب توضیحی اســـت که از رویکرد نظریه‌بازی‌ها استفاده می‌کند و به ما کمک می‌کند بفهمیم چرا یک مدل یادگیری ماشین خروجی خاصی را تولید می‌کند. الگوریتم SHAP، در حالی که برای ارایه توضیحات قابـــل تفسیر در یادگیری ماشین قدرتمند اســـت، با چندین چالش روبه‌رو اســـت. مشابه با الگوریتم LIME، این الگوریتم نیز از نظر محاســـباتی پرهزینه است، به‌ویـــژه برای مدل‌هایی بـــا ویژگی‌های زیاد یـــا داده‌های بزرگ، زیرا نیاز به محاسبه مقادیر شاپلی برای هر ترکیب ممکـــن از ویژگی‌ها دارد. SHAP همچنیـــن در فضاهای با ابعـــاد بالا که تعـــداد ترکیب‌های ممکـــن ویژگی‌ها به طور نمایی رشد می‌کند، با مقیاس‌پذیری مشکل دارد. وابستگی SHAP به نظریه بازی ممکن است منجر به ناسازگاری در

22-Local Interpretable Model-Agnostic Explanations
23-SHapley Additive exPlanation

کیفیت توضیحات شـــود، به‌ویژه برای مدل‌های پیچیده و غیرخطی. علاوه بر این، مقادیر شاپلی به نقطه مرجع یا پایه انتخاب شده حساس هستند و انتخاب پایه مناسب می‌تواند تأثیر زیادی بر تفسیر داشـــته باشـــد. در نهایت، در حالی که SHAP بینش‌هـــای کلی را ارایه می‌دهد، تفسیر خروجی آن بـــرای مدل‌های بسیار پیچیده همچنـــان می‌تواند برای کاربران غیرمتخصص به دلیل پیچیدگی توضیحات تولید شده چالش برانگیز باشد.

از زمـــان معـــرفی XAI، ایـــن حـــوزه توجـــه بسیاری از پژوهشـــگران را بـــه خـــود جلب کرده اســـت. مدل‌های توضیح‌دهنـــده مختلـــفی پیشـــنهاد شـــده‌اند و ایـــن توضیح‌دهنده‌ها با موفقیت به مســـایل عملی مختلف اعمال شده‌اند. با این که حوزه‌های کاربردی مختلفی توسط XAI پوشش داده شده‌اند، تعداد کمی از کارها به‌طور هم‌زمان به XAI و مهندسی نرم‌افزار یا تحلیل نیازمندی‌های نرم‌افزاری پرداخته‌اند.

حبیبـــه و همکاران بحثی درباره هم‌افـــزایی بین XAI و مهندسی نیازمندی‌ها ارایه کردنـــد [۱۱]. آن‌ها چالش‌ها را بیان کرده و چارچوبی برای مقابله با آن‌ها پیشنهاد کردند. چالش‌های اصلی شـــامل نبود نقش واســـطه، عدم وجود تعریف منسجم از توضیح‌پذیری، کمبود روش‌های متمرکز بر سودبران، و عدم وجود واژگان مشترک است. چارچوب پیشـــنهادی برای مقابله با این چالش‌ها شـــامل پنج مرحله است که با شناسایی ســـودبران آغاز شده و با طبقه‌بندی نیازمندی‌ها پایان می‌یابد.

چازت و همکاران در [۲۴] ذکر کرده‌اند که توضیح‌پذیری در حوزه نیازمندی‌های نرم‌افزار مورد مطالعه قرار نگرفته اســـت و محدودیت‌هایی برای اعمال توضیح‌پذیری در این زمینه وجـــود دارد. آنها راهکاریی را جهـــت رفع موانع و محدودیت‌هـــای موجود با تمرکز بـــر توضیح‌پذیری، ارایه یـــک مدل مفهومی، کاتالوگ دانـــش و یک مدل مرجع برای سیستم‌های توضیح‌پذیر ارایه داده‌اند. در پژوهش دیگری، بهره‌گیـــری از هـــوش مصنوعی قابل توضیـــح در مرحلهٔ مهندسی نیازمندی‌ها مورد مطالعه قرار گرفته اســـت [۲۵]. مگباله در این پژوهش بیان کرده است هوش مصنوعی قابل توضیح هنوز به‌طور گسترده در مرحله مهندسی نیازمندی‌ها استفاده نشده است، زیرا هیچ رویکرد استانداردی پیشنهاد نشده است. هیچ یک از مقالات بررسی شده توسط ایشان به‌طور واقـــعی از هوش مصنوعی و زیرتکنیک‌های مرتبط با آن برای کار با جنبه‌گرایی اســـتفاده نکرده‌اند و همچنین توجه کمتری به ادبیات مربوط به اســـتفاده از تکنیک‌های هوش مصنوعی در مرحله اعتبارسنجی نیازمندی‌ها شده اســـت، در حالی که این مرحله یـــکی از مراحل بسیار مهم در کل فرآیند مهندسی نیازمندی‌ها محســـوب می‌شـــود و استفاده از هوش مصنوعی بیشتر بر استخراج نیازمندی‌ها و مدیریت نیازمندی‌ها متمرکز بوده است. بای و همکاران از ابزار تحلیل احساسات، مدل‌سازی موضوعی ساختاری، ابـــزار تحلیل وب، یادگیـــری ماشیـــن و تکنیک‌های هوش مصنوعی قابل توضیح برای تحلیل عمیق نیازهای کاربران در ســـه برنامه کاربردی ویدئو کنفرانس استفاده کرده‌اند. آنهـــا از هوش مصنوعی قابل توضیح برای اولویت‌بندی و رتبه‌بندی نیازمندی‌های نرم‌افزار استفاده کرده‌اند [۲۶].

کلمان و همکاران مروری بر هم‌راستایی XAI با فرآیند توســـعه نرم‌افزار ارایه کردنـــد [۱۲]. آن‌ها مراحل اصلی فرآیند توسعه نرم‌افزار مانند مهندسی نیازمندی‌ها، تحلیل و طـــراحی را در نظر گرفتند و روش‌هـــا و ابزارهای XAI را با این مراحل توســـعه هم‌راستا کردند. هدف آن‌ها ارایه یک نقشـــه راه برای توســـعه برنامه‌های هوش مصنوعی با اســـتفاده از توضیح‌پذیری بود که چرخه عمر کلاسیک توسعه سیســـتم‌های نرم‌افزاری را در نظر می‌گرفت. آنها معتقدند که چالش چنین روش‌هایی نحوه انتخاب تنظیمات بهینه، مانند تعداد نمونه‌های اولیه است و چالش اصلی این اســـت که توضیحات برای انســـان طراحی شده است، در نتیجه ارزیابی چنیـــن توضیحاتی را تا حدی ذهنی می‌کند. دریافت‌کنندگان توضیحات می‌توانند با توجه به انتظارات، تخصص یادگیری ماشین و دانش زمینه‌ای بسیار متفاوت

باشند. بنابراین، یک توضیح یکسان می‌تواند برای یک کاربر راضی‌کننده باشد اما برای کاربر دیگر کاملاً طاقت‌فرسا یا غیرقابل درک باشد.

یک نظرسنجی کیفی برای بررسی درک کارشناسان از اهداف مدل‌های پیش‌بینی نقض و تکنیک‌های مدل‌محور مستقل که برای توصیف بصری مدل‌های پیش‌بینی نقض استفاده می‌شوند، در [۱۳] ارایه شده است. این کار نشان داد که تکنیک‌های مدل‌محور مستقل مانند LIME به کارشناسان کمک می‌کنند تا مدل‌های پیش‌بینی نقض و پیش‌بینی‌های آن‌ها را درک کنند.

با وجود استفاده موثر از الگوریتم‌های LIME و SHAP در مسایل پردازش زبان طبیعی این الگوریتم‌ها با چالش‌هایی در این زمینه روبه‌رو هستند. به دلیل ابعاد بالای داده‌های متنی، وابستگی‌های ناشی از ماهیت زبان مورد پردازش و پیچیدگی مدل‌های پردازش زبان طبیعی این چالش‌ها به‌وجود می‌آیند. هر دو الگوریتم در ایجاد تغییرات معنادار در متن مشکل دارند، زیرا تغییر کلمات می‌تواند به طور چشمگیری معنای جمله را تغییر دهد و درک روابط وابسته به زمینه دشوار است. تعداد زیاد ویژگی‌ها (کلمات) در مدل‌های پردازش زبان طبیعی بار محاسباتی را برای هر دو الگوریتم افزایش می‌دهد و آن‌ها را از نظر محاسباتی پرهزینه می‌کند. علاوه بر این، این الگوریتم‌ها ممکن است توضیحاتی برای مدل‌های پیچیده مانند ترنسفورمرها ایجاد کنند که درک آنها ساده نباشد. هر دو روش همچنین با حساسیت به حملات خصمانه، ناهماهنگی‌های پیش‌پردازش و مقیاس‌پذیری روبه‌رو هستند که می‌تواند باعث ایجاد اشکالاتی در کارایی آنها شود.

۳- هوش مصنوعی قابل توضیح

هوش مصنوعی قابل توضیح یک مفهوم نوظهور در حوزه هوش مصنوعی است که به طور خاص بر افزایش قابلیت توضیح‌پذیری در راه‌حل‌های هوش مصنوعی تمرکز دارد. یک مدل قابل توضیح باید برای کاربران انسانی قابل فهم، قابل‌اعتماد و توجیه‌پذیر باشد و شفافیت را در راه‌حل‌های پیچیده هوش مصنوعی فراهم کند. این مفهوم به دنبال حل چالش‌های مربوط به درک و اعتماد کاربران به مدل‌های هوش مصنوعی، به‌ویژه در سیستم‌های پیچیده، است. اهمیت هوش مصنوعی قابل توضیح در کاربردهایی که تصمیمات مدل تأثیر مستقیم بر زندگی انسان‌ها دارند، مانند حوزه‌های پزشکی، مالی و حقوقی، به‌طور چشمگیری افزایش یافته است. یک مدل قابل توضیح باید نه تنها برای کاربران انسانی قابل فهم باشد، بلکه باید اعتماد کاربران را نیز جلب کند. همچنین، توجیه‌پذیری نتایج مدل‌ها و ایجاد شفافیت در فرآیند تصمیم‌گیری از الزامات کلیدی هوش مصنوعی قابل توضیح هستند. شفافیت، به معنای فراهم کردن بینش در مورد چگونگی عملکرد مدل‌ها و دلایل اتخاذ یک تصمیم خاص، به کاربران این امکان را می‌دهد که مدل‌ها را بهتر درک کنند و در صورت لزوم عملکرد آن‌ها را بهبود بخشند. هوش مصنوعی قابل توضیح دو هدف را دنبال می‌کند [۱۴]:

ایجاد راه‌حل‌های دقیق و کارآمد: بهبود دقت و عملکرد مدل‌های هوش مصنوعی بدون قربانی کردن قابلیت توضیح‌پذیری.

ارایه توضیحات شفاف و واضح: تضمین این که نتایج پیش‌بینی شده برای کاربران قابل فهم باشند و بتوان به آن‌ها اعتماد کرد.

برای دستیابی به این اهداف روش‌های مختلف هوش مصنوعی قابل توضیح ارایه شده‌اند که به سه دسته قابل تقسیم هستند:

روش‌های تفسیرپذیری: تکنیک‌های تفسیر مدل که به دنبال درک عملکرد داخلی مدل‌های هوش مصنوعی و شناسایی ویژگی‌هایی هستند که در پیش‌بینی‌ها بیشترین اهمیت را دارند.

روش‌های تولید توضیح: این روش‌ها برای توضیح واضح پیش‌بینی مدل‌های هوش مصنوعی استفاده می‌شوند.

روش‌های تحلیل حساسیت: این روش‌ها نشان می‌دهند

کـــه چگونه پیش‌بینی مدل تحت تأثیـــر تغییرات کوچک در داده‌های ورودی قرار می‌گیرد.

در ســـال‌های اخیر مدل‌های توضیحی مختلفی توسط پژوهشـــگران پیشنهاد شـــده اســـت که LIME و SHAP از محبوب‌ترین آن‌ها هستند [۱۵]. به دلیل تعداد زیاد مدل‌های توضیح‌دهنـــده، انتخاب یک توضیح‌دهنده مناســـب ممکن اســـت یک کار چالش‌برانگیز باشد. ژیزوس و همکاران یک مدل آزمایشی پیشـــنهاد دادند که به کاربر در انتخاب مدل مناسب کمک می‌کند [۱۵].

۳-۱- توضیح الگوریتم LIME

در این مقاله از روش LIME استفاده شده است که یکی از پرکاربردترین و محبوب‌ترین تکنیک‌های توضیح‌پذیری مدل‌های یادگیری ماشین در سال‌های اخیر است. این روش با هدف ارایه توضیحات محلی برای پیش‌بینی‌های مدل‌های یادگیری ماشین طراحی شده و قابلیت اعمال بر روی انواع مختلف مدل‌ها را دارد. از آنجایی که LIME مستقل از مدل اســـت، می‌تواند برای توضیح پیش‌بینی‌های الگوریتم‌های گوناگـــون از جمله طبقه‌بندی و رگرسیون مورد اســـتفاده قرار گیرد. روش کار LIME به این صورت اســـت که ابتدا داده‌های ورودی را تغییـــر داده و تأثیر این تغییرات را بر پیش‌بینی‌هـــای مدل تحلیل می‌کنـــد. برای یک نمونه خاص، LIME مجموعه‌ای از نمونه‌های تصادفی را در اطراف داده ورودی اصلی تولید می‌کند و مدل اصلی را برای پیش‌بینی این نمونه‌ها مورد استفاده قرار می‌دهد. سپس، با استفاده از پیش‌بینی‌های مدل برای این نمونه‌ها، یک مدل ســـاده و قابل فهم (مانند یک مدل خطی) ایجاد می‌کند که رفتار مدل اصلی را در اطراف داده ورودی توضیح می‌دهد. این مدل ســـاده به صورت محلی عمل می‌کند، بـــه این معنا که تنها پیش‌بینی مربوط به نمونه ورودی اصلی را توضیح می‌دهد و نه کل مجموعه داده یا ساختار کلی مدل.

یکی از ویژگی‌های برجسته LIME، توانایی آن در ارایه توضیحات محلی دقیق است. این ویژگی به‌خصوص برای مدل‌های پیچیـــده و جعبه سیاه، مانند شـــبکه‌های عصبی عمیق یا مدل‌های تقویتی، بسیار مفید است. چنین مدل‌هایی به دلیل ساختار داخلی پیچیده، معمولاً غیرشفاف هستند و کاربران نمی‌توانند به ســـادگی علت پیش‌بینی‌های آن‌ها را درک کنند. LIME این چالش را با تولید توضیحات قابل فهم برای پیش‌بینی‌های خاص برطرف می‌کند و اعتماد کاربران را به نتایـــج مدل افزایش می‌دهد. علاوه بـــر این، LIME از نظر پیاده‌ســـازی نیز انعطاف‌پذیری بالایی دارد. این روش می‌توانـــد بـــر روی طیف وسیـــعی از مدل‌هـــای یادگیری ماشین اعمال شـــود و برای کاربران امکان تحلیل و درک نتایج پیش‌بیـــنی را فراهم می‌کند. ابزارهایی مانند LIME به پژوهشـــگران و توســـعه‌دهندگان این امکان را می‌دهند که علاوه بر بهبود دقت مدل‌های خود، شفافیت و توضیح‌پذیری آن‌ها را نیز ارتقا دهند. این امر به ویژه در زمینه‌هایی مانند پزشکی، مالی و امنیت سایبری که اعتماد به مدل‌های هوش مصنوعی ضروری است، اهمیت بسیاری دارد. به‌طورکلی، LIME به عنوان یک ابزار قدرتمند و مستقل از مدل، روشی قابل اعتمـــاد برای درک پیش‌بینی‌هـــای مدل‌های یادگیری ماشین ارایـــه می‌دهد و نقشی کلیدی در ترویج اســـتفاده مسئولانه و شفاف از هوش مصنوعی ایفا می‌کند.

الگوریتم LIME می‌تواند به صورت زیر مدل شود. برای ســـادگی، جزئیات در اینجا نادیده گرفته شده است. فرض کنیـــد که یک نمونه ورودی x داریم و می‌خواهیم پیش‌بینی ارایه شـــده توســـط طبقه‌بند C را توضیح دهیم. پیش‌بینی انجام شده توســـط مدل $y = C(X)$ است. الگوریتم LIME نمونه‌هـــای n تایی $x' = \{x'_1.x'_2.\dots.x'_n\}$ را با تغییر دادن نمونه اصلی تولید می‌کند.

خروجی $y'_i = C(x'_i)$ برای نمونه x'_i توسط الگوریتم جعبه سیاه تولید و نمونه برچســـب‌گذاری می‌شـــود. این نمونه‌ها به‌عنوان یک مجموعه داده برای آموزش مدل قابل توضیح استفاده می‌شوند. سپس، الگوریتم LIME نمونه‌ها را بر اساس نزدیکی به نمونه اصلی x وزن‌دهی می‌کند. به این ترتیب، نمونه‌های نزدیک‌تر به x وزن بیشـــتری دارند.

وزن هر نمونه x_i' با استفاده از هسته گاوسی[۲۴] به صورت زیر است:

$$w(x_i') = exp\left(-\frac{d(x.x_i')^{۲}}{۲\sigma^{۲}}\right)$$

که در آن $d(x.x_i')$ فاصله اقلیدسی بین x و x' است و σ میزان گسترش هسته را کنترل می‌کند. پس از آن، میانگین وزنی پیش‌بینی‌های y_i' محاسبه می‌شود. سپس، پیش‌بینی‌های وزنی برای تطبیق یک مدل قابل توضیح g استفاده می‌شوند. LIME از مدل‌های ذاتی قابل توضیح مانند رگرسیون خطی، درخت تصمیم‌گیری و مدل‌های ذهنی قاعده‌محور[۲۵] برای توضیح نتایج استفاده می‌کند. فرض کنید که از رگرسیون خطی استفاده می‌شود. مدل قابل توضیح g برای نمونه‌های تغییر یافته x_i' به صورت زیر برازش می‌شود:

$$g(x_i') = \theta_{.} + \sum_{i=۱}^{k} \theta_i x_i'$$

که در آن θ_i ضریب ویژگی i ام در رگرسیون خطی است. در نهایت، مدل تولید شده g برای توضیح پیش‌بینی انجام‌شده توسط مدل C برای ورودی اصلی x استفاده می‌شود. توضیح الگوریتم LIME مجموعه‌ای از امتیازهای اهمیت ویژگی است. هر وزن نشان می‌دهد که یک ویژگی در x چقدر به پیش‌بینی کمک می‌کند. برای رگرسیون خطی، ضرایب θ به‌عنوان وزن‌هایی در نظر گرفته می‌شوند که توسط الگوریتم LIME گزارش می‌شوند.

۴- روش پیشنهادی

با توجه به کاربردهای موفق XAI در زمینه‌های مختلف، در این مقاله تلاش شده تا توضیح‌دهنده LIME به مشکل طبقه‌بندی نیازمندی‌ها اعمال شود. هدف این مطالعه این است که نحوه استفاده از XAI در مرحله توسعه نرم‌افزار برای اعضای تیم توسعه توضیح داده شود، همچنین نتایج اعمال XAI در مرحله مهندسی نیازمندی‌ها برای تیم توسعه نرم‌افزار مشخص شود. بدین‌منظور یک روش که شامل یک طبقه‌بند یادگیری ماشین و توضیح‌دهنده LIME است، برای تحلیل نیازمندی‌های نرم‌افزاری استفاده شده است. ساختار روش پیشنهادی در شکل ۱ ارایه شده است. این شکل نشان می‌دهد که ساختار دارای دو بخش اصلی است. بخش اول به طبقه‌بند اختصاص دارد و بخش دوم متعلق به توضیح‌دهنده است. برای استفاده از مدل XAI همراه با یک طبقه‌بند، ابتدا طبقه‌بند طراحی می‌شود. سپس، مدل XAI می‌تواند برای توضیح نمونه‌های مورد نیاز استفاده شود.

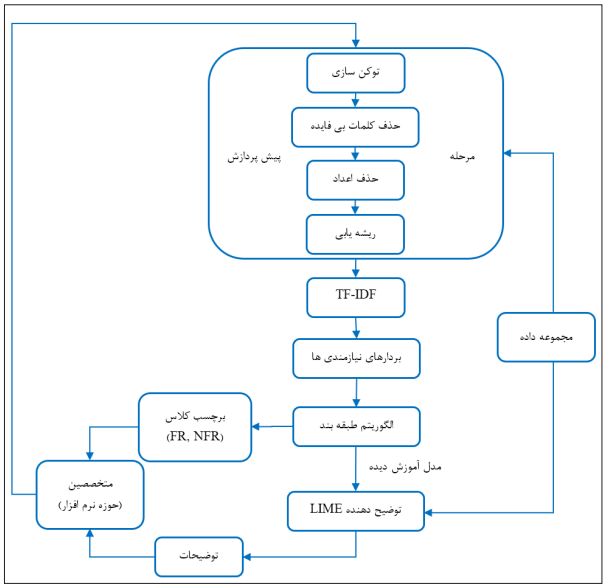
این روش یک مجموعه نیازمندی به عنوان ورودی دریافت می‌کند. هر رکورد در مجموعه داده دارای دو ستون است که ستون اول شامل یک جمله یا پاراگراف از یک نیازمندی و ستون دوم برچسب کلاس است. نیازمندی‌ها به زبان طبیعی هستند. بنابراین، داده‌ها باید پیش‌پردازش شوند. مرحله پیش‌پردازش شامل برخی مراحل شناخته شده است که به طور عمومی در برنامه‌های NLP[۲۶] استفاده می‌شوند. وظایف مهم شامل توکن‌سازی، حذف اعداد و کلمات بی‌فایده و ریشه‌یابی است.

علاوه بر این مراحل، یک مرحله جدید اضافه می‌شود که به کارشناس حوزه این امکان را می‌دهد که پردازش خاص برنامه‌کاربردی را روی خروجی مرحله ریشه‌یابی، اعمال کند. دلیل اضافه کردن چنین مرحله‌ای در بخش بعدی توضیح داده می‌شود. به عنوان مثال، کلمات کیفی مانند «باید[۲۷]»، «می‌بایست[۲۸]» و «لازم است[۲۹]» معمولاً برای تعیین اولویت نیازمندی‌ها استفاده می‌شوند و به نظر می‌رسد که ممکن است در طبقه‌بندی نیازمندی‌ها نادیده گرفته شوند. بنابراین، این کلمات می‌توانند از نیازمندی‌ها حذف شوند. همچنین، نام نرم‌افزار ممکن است اطلاعات اضافی برای طبقه‌بندی نیازمندی‌ها ارایه نکند.

خروجی مرحله پیش‌پردازش یک مجموعه بردار است که در آن هر بردار شامل ریشه‌های کلمات مهم در یک

24-Gaussian kernel
25-Rule-Based Heuristic (RBH)

26-Natural Language Processing
27-Shall
28-Should
29-Must



شکل ۱. ساختار روش پیشنهادی برای تحلیل نیازمندی‌ها با استفاده از هوش مصنوعی قابل توضیح و طبقه‌بند SVM

نیازمندی همراه با برچسب کلاس آن است. این بردارها به عنوان ورودی برای مرحله بعدی طبقه‌بندی استفاده می‌شوند.

بیشتر طبقه‌بندها با داده‌های متنی به عنوان ورودی کار نمی‌کنند. بنابراین، داده‌های متنی معمولاً به بردارهای عددی با استفاده از روش‌های مختلف تبدیل می‌شوند. در اینجا، بر اساس تحقیقات قبلی [۱۶]، [۱۷] از روش TF-IDF برای تبدیل نیازمندی‌ها به بردارهای عددی استفاده کردیم. پس از اعمال روش TF-IDF، از طبقه‌بند برای طبقه‌بندی نیازمندی‌ها استفاده می‌شود. با آموزش طبقه‌بند، مدل تولید شده می‌تواند توسط مدل XAI برای توضیح این که چرا یک نیازمندی به عنوان کارکردی یا غیرکارکردی طبقه‌بندی می‌شود، استفاده شود.

در اینجا از مدل توضیح‌دهنده LIME استفاده شده است. نیازمندی که می‌خواهیم آن را توضیح دهیم و طبقه‌بندی تولید شده را به عنوان ورودی به LIME داده و توضیح مربوط به آن تولید می‌شود. برای داده‌های متنی مانند نیازمندی‌های کاربران، مدل LIME مجموعه‌ای از کلمات حمایتی[۳۰] و مختل‌کننده[۳۱] را همراه با وزن‌های آن‌ها در توضیح و احتمال پیش‌بینی کلاس نیازمندی تولید می‌کند.

در نهایت، خروجی مدل XAI همراه با کلاس پیش‌بینی شده توسط طبقه‌بند می‌تواند توسط کارشناسان دامنه مانند تحلیل‌گران، مشخص‌کنندگان نیازمندی یا معماران، به‌عنوان بخشی از کارهایشان استفاده شود. سپس کارشناس دامنه برچسب‌های کلاس پیش‌بینی شده را همراه با توضیحات تولید شده توسط الگوریتم LIME تحلیل می‌کند. نتیجه این تحلیل می‌تواند به عنوان بازخورد برای مرحله پیش‌پردازش استفاده شود.

متد پیشنهادی به کارشناسان دامنه در فرآیند توسعه نرم‌افزار کمک می‌کند تا ویژگی‌های مؤثر در پیش‌بینی کلاس یک نیازمندی را مشاهده کنند. بنابراین، کارشناسان دامنه می‌توانند به خروجی مدل‌های یادگیری ماشین در تحلیل نیازمندی‌ها اعتماد کنند.

۵- مطالعه تجربی

برای بررسی قابلیت کاربرد روش پیشنهادی که شامل ماژول XAI برای تحلیل نیازمندی‌های نرم‌افزاری است، از مجموعه داده PROMISE استفاده شده است. این مجموعه داده شامل نیازمندی‌های کارکردی و غیرکارکردی است. نیازمندی‌های کارکردی با برچسب (F) و نیازمندی‌های غیرکارکردی به یازده دسته تقسیم می‌شوند که برچسب‌های زیر را دارند: دسترس‌پذیری (A)[۳۲]، تحمل خطا (FT)[۳۳]، قانونی (L)[۳۴]، ظاهر و احساس (LF)[۳۵]، قابلیت نگهداری (MN)[۳۶]، عملیاتی (O)[۳۷]، عملکرد (PE)[۳۸]، قابلیت حمل (PO)[۳۹]، مقیاس‌پذیری (SC)[۴۰]، امنیت (SE)[۴۱] و کاربردپذیری (US)[۴۲]. در این کار، طبقه‌بندی نیازمندی‌ها به عنوان یک مسئله طبقه‌بندی دوتایی در نظر گرفته شده است. بنابراین، تمام زیرمجموعه‌های

30-Supportive
31-Distractive

32-Availability
33-Fault Tolerance
34-Legal
35-Look & Feel
36-Maintainability
37-Operational
38-Performance
39-Portability
40-Scalability
41-Security
42-Usability

جدول ۱. عملکرد ماشین بردار پشتیبان در مجموعه آزمایشی

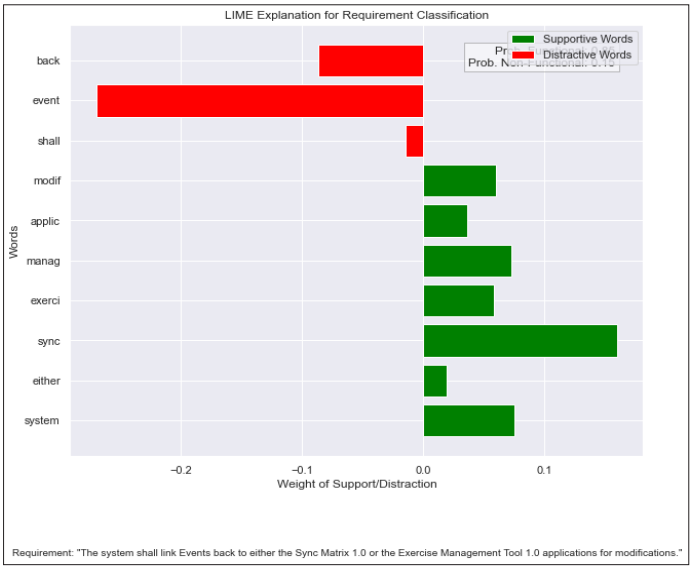
معیار	مقدار
F1-score	٪ ۸۹
دقت (Accuracy)	٪ ۹۰
دقت (Precision)	٪ ۸۸
یادآوری (Recall)	٪ ۹۱

نیازمندی‌های غیرکارکـــردی به‌عنوان غیرکارکردی در نظر گرفته می‌شوند. به‌طور کلی، ٤٠ درصد نیازمندی‌ها کارکردی و ٦٠ درصـــد غیرکارکردی هســـتند. مجموعـــه داده به یک مجموعه آموزش و یک مجموعه تست تقسیم شده است که ۸۰ درصد برای آموزش و باقیمانده برای آزمایش استفاده می‌شـــود. نمونه‌ها در مجموعه‌های آمـــوزشی و آزمایشی به‌صورت تصادفی انتخاب شـــده‌اند. این روش با اســـتفاده از زبان برنامه‌نویسی پایتون پیاده‌ســـازی شد. از بسته‌های Pandas، Lime، Keras، NLTK، NumPy و Seaborn استفاده شده است.

الگوریتم LIME می‌تواند با هر روش طبقه‌بندی کار کند. بنابراین، هر الگوریتم یادگیـــری ماشین یا یادگیری عمیق می‌تواند به عنوان ماژول طبقه‌بندی در شـــکل ۱ قرار گیرد. در اینجـــا، از SVM به عنوان الگوریتم طبقه‌بندی اســـتفاده شـــده اســـت. عملکرد طبقه‌بند در داده‌هـــای آزمایشی در جدول ۱ ارایه شـــده اســـت. نتایج میانگین پـــس از ۳۰ بار اجرای این روش در این جدول گزارش شـــده است. نتایج نشـــان می‌دهد که ماشین‌بردار پشتیبان (Support Vector Machine) عملکرد قابل قبولی در داده‌های آزمایشی دارد

۵-۱- خروجی توضیح‌دهنده

قبل از تحلیل قابلیت کاربرد الگوریتم LIME برای تحلیل نیازمندی‌هـــا، خروجی LIME در اینجا ارایه شـــده اســـت. راه‌حلی که توسط LIME تولید می‌شود شامل مجموعه‌ای از کلمات است که تصمیم را پشتیبانی یا مختل می‌کند. یک مثال از خروجی LIME در شکل ۲ آورده شده است. محور عمودی لیست کلمات حمایتی و کلمات مختل‌کننده را نشان



شکل ۲. خروجی توضیح‌دهنده LIME بر روی یک نیازمندی انتخاب‌شده به‌صورت تصادفی در مجموعه داده PROMISE

The system shall link Events back to either the Sync Matrix 1.0 or the Exercise Management Tool 1.0 applications for modifications.

شکل ۳. نیازمندی مربوط به توضیح شکل ۲

می‌دهد. کلمات حمایتی و مختل‌کننده به ترتیب با رنگ‌های ســـبز و قرمـــز نمایش داده می‌شـــوند. محـــور افقی وزن حمایت یا اختلال کلمات را نشـــان می‌دهد. همچنین،LIME احتمال در نظر گرفتن یک نیازمندی به عنوان کارکردی یا غیرکارکردی را ارایه می‌دهد.

نیازمندی مربوطه در شـــکل ۳ ارایه شـــده است. متن نیازمندی پس از پیش‌پردازش در اینجا نشـــان داده شـــده اســـت. این یک نیازمنـــدی غیرکارکردی اســـت و احتمال غیرکارکردی بودن آن توســـط LIME بیشتر از ۰/۹ است. این شکل نشـــان می‌دهد که برخی از کلمات مانند «لینک»، «ابزار» و «ماتریس» برای طبقه‌بندی این نیازمندی در نظر گرفته نشده‌اند.

۵-۲- قابلیت توضیح مدل

این آزمایـــش به‌منظور تحلیل توضیـــح پیش‌بینی‌های

جدول ۲. تعریف کلمات حمایتی و مختل‌کننده

معیار	تعریف
کلمات حمایتی (S)	مجموعه‌ای از تمام کلمات با اثر مثبت در طبقه‌بندی یک نیازمندی به‌عنوان نیازمندی غیرکارکردی.
کلمات مختل‌کننده (D)	مجموعه‌ای از تمام کلمات با اثر منفی در طبقه‌بندی یک نیازمندی به‌عنوان نیازمندی غیرکارکردی.

نیازمندی‌های غیرکارکردی در مجموعه داده با استفاده از الگوریتم LIME طراحی شده است. به‌نظر می‌رسد خروجی توضیح‌دهنده که در شکل ۲ نشان داده شده است، بینش ارزشمندی برای ما فراهم می‌کند تا قابلیت کاربرد الگوریتم LIME را تحلیل کنیم. به‌عنوان مثال، کلمه «استفاده» که یک کلمه حمایتی قوی است، برای در نظر گرفتن این نیازمندی به‌عنوان غیرکارکردی استفاده می‌شود. نتایجی که در اینجا و در بخش بعدی گزارش شده است، تنها مجموعه‌های آزمایشی را در نظر می‌گیرد.

برای تحلیل قابلیت کاربرد LIME برای تحلیل نیازمندی‌ها، از دو مجموعه کلمه که توسط الگوریتم LIME گزارش شده است، برای تعیین نیازمندی‌ها به‌عنوان غیرکارکردی استفاده می‌شود. در اینجا، ما فقط توضیح یک نیازمندی به‌عنوان غیرکارکردی را در نظر می‌گیریم. سناریوی مشابهی می‌تواند برای نیازمندی‌های کارکردی استفاده شود. این کلمات به‌عنوان کلمات حمایتی و مختل‌کننده شناخته می‌شوند. تعاریف کلمات حمایتی و مختل‌کننده در جدول ۲ ارایه شده است.

سه آزمایش برای نشان دادن قابلیت توضیح الگوریتم LIME برای نیازمندی‌های نرم‌افزاری انجام شده است. در آزمایش اول، تمام کلمات حمایتی و مختل‌کننده در نیازمندی‌ها در نظر گرفته می‌شوند. برای این منظور، داده‌های آزمایشی به‌صورت نمونه به الگوریتم LIME داده می‌شود و کلمات حمایتی و مختل‌کننده جمع‌آوری می‌شوند. تمام این کلمات تجمیع شده و رخداد آنها محاسبه می‌شود

شکل‌های ٤ و ٥ کلمات حمایتی و مختل‌کننده را برای توضیح طبقه‌بندی یک نیازمندی به‌عنوان غیرکارکردی نشان می‌دهند. محور افقی نشان‌دهنده کلماتی است که در غیرکارکردی‌ها ظاهر شده‌اند و محور عمودی درصد وقوع این کلمات را نشان می‌دهد. درصد وقوع به‌صورت زیر تعریف می‌شود:

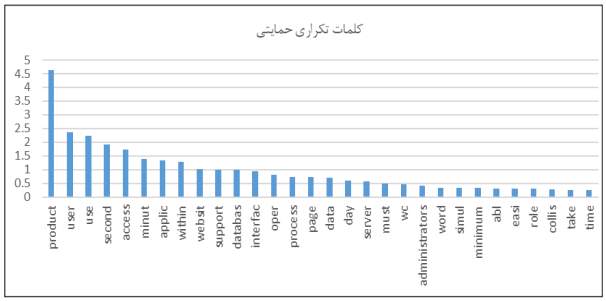
$$pc_i = \frac{n_i}{N} \times ۱۰۰$$

در این فرمول n_i تعداد وقوع کلمه i است و N تعداد کل کلمات می‌باشد. الگوریتم LIME ده‌ها کلمه حمایتی و مختل‌کننده را برای توضیح تمام نمونه‌های آزمایشی گزارش کرده است. نمایش تمام آن‌ها ممکن نیست. به همین دلیل، تنها ۳۰ کلمه برتر با بالاترین درصد وقوع در اینجا نمایش داده شده است. ریشه‌های کلمات در این شکل‌ها ارایه شده است.

همان‌طور که در شکل ٤ نشان داده شده است، «محصول»، «کاربر»، «استفاده»، «ثانیه» و «دقیقه» پنج کلمه حمایتی با بالاترین تکرار هستند. اگرچه این کلمات در پیش‌بینی یک نیازمندی به‌عنوان غیرکارکردی بسیار تکرار شده‌اند، اما در میان کلمات مهم‌ترین نیستند. معمولاً یک کارشناس انسانی نمی‌تواند تنها با استفاده از این کلمات، یک نیازمندی غیرکارکردی را تعیین کند و برای طبقه‌بندی یک نیازمندی به‌عنوان غیرکارکردی، به کلمات دقیق‌تری نیاز دارد. احتمالاً در مدل یادگیری ماشین نیز همین وضعیت وجود دارد. به‌نظر می‌رسد که کلمات خاص غیرکارکردی توسط روش‌های یادگیری ماشین برای طبقه‌بندی استفاده می‌شود.

همچنین کلمات «بازیکن»، «اجازه»، «اختلاف»، «هدایت» و «حرکت» پنج کلمه مختل‌کننده با بالاترین تکرار هستند که در شکل ٥ نشان داده شده‌اند. برخی از کلمات با تکرار بالا مانند «کلمه»، هم در کلمات حمایتی و هم در کلمات مختل‌کننده مشاهده می‌شود.

برای تفکیک و تحلیل بهتر کلمات، تمام کلمات استفاده



شکل ۴. لیست تکراری‌ترین کلمات حمایتی در طبقه‌بندی یک نیازمندی به‌عنوان غیرکارکردی در مجموعه داده قبل از حذف کلمات کیفی

شده توسط توضیح‌دهنده برای توضیح طبقه‌بند SVM را به سه مجموعه تقسیم کردیم. این مجموعه‌ها A،B و C نامیده می‌شوند و به شرح زیر تعریف شده‌اند:

A=D−C مجموعه کلمات مختل‌کننده منهای کلمات مشترک.

B=S−Cمجموعه کلمات حمایتی منهای کلمات مشترک

C=S∩Dمجموعه کلمات مشترک که تقاطع مجموعه‌های S و D است.

بعضی از اعضای هر مجموعه عبارتند از:

$$A = \{player, allow, dispute., diplay, meet, \dots\}$$

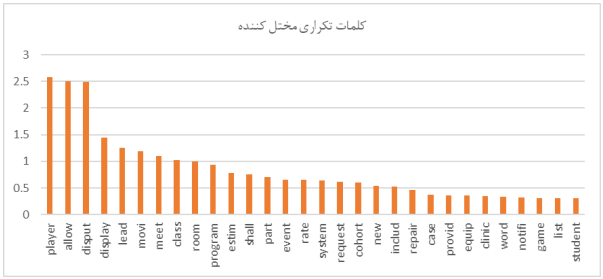
$$B = \{support, second, minute., server, \dots\}$$

$$C = \{word, shall, system, user, \dots\}$$

مجموعه کلمات حمایتی (S) و مجموعه کلمات مختل‌کننده (D) در جدول ۲ تعریف شده‌اند.

این دسته‌بندی کلمات نشان می‌دهد که توضیح‌دهنده به‌طور معقولی از آن‌ها برای پیش‌بینی یک نیازمندی به‌عنوان کارکردی یا غیرکارکردی استفاده می‌کند. مجموعه‌های A و B از مجموعه C مهم‌تر هستند. با این حال، کلمات مشترک نیاز به توجه بیشتری دارند.

مجموعه C از کلمات مشترک، کلماتی مانند «باید»، «بایستی»، «کاربر» تشکیل شده است. اگرچه این کلمات به‌طور مکرر تکرار شده‌اند، به‌نظر می‌رسد که چنین کلماتی نمی‌توانند به‌طور قابل‌توجهی توسط یک طبقه‌بند برای طبقه‌بندی یک نیازمندی به‌عنوان کارکردی یا غیرکارکردی استفاده شوند. همین وضعیت در دنیای واقعی وجود دارد



شکل ۵. کلمات تکراری مختل‌کننده در طبقه‌بندی یک نیازمندی به‌عنوان غیرکارکردی در مجموعه داده قبل از حذف کلمات کیفی

و یک کارشناس انسانی نمی‌تواند تنها به کلمات مشترک برای طبقه‌بندی یک نیازمندی نرم‌افزاری تکیه کند.

مجموعه A شامل کلمات مختل‌کننده است، کلماتی مانند «نمایش»، «تغییر»، «کلاس»، «حذف» و غیره. معمولاً، این کلمات برای نشان دادن مسئولیت یا خدماتی که توسط نرم‌افزار ارایه می‌شود، استفاده می‌شوند. این کلمات می‌توانند به‌عنوان نامزدهایی برای تعیین یک نیازمندی به‌عنوان غیرکارکردی در نظر گرفته شوند. بنابراین، به‌نظر می‌رسد که توضیح‌دهنده این کلمات را به‌عنوان کلمات مختل‌کننده شناسایی کرده است. باید توجه داشت که به‌دلیل خطاهای طبقه‌بندی، دسته‌بندی کلمات در این مجموعه‌ها ممکن است شامل خطا باشد.

مجموعه B که شامل کلمات حمایتی است، شامل کلماتی مانند «ثانیه»، «دقیقه»، «نصب»، «مجوز»، «تهدید» و غیره می‌شود که برای طبقه‌بندی نیازمندی‌ها به‌عنوان غیرکارکردی معقول هستند. این کلمات جالب توجه هستند و توسط طبقه‌بند برای طبقه‌بندی یک نیازمندی به‌عنوان غیرکارکردی استفاده می‌شوند. به‌عنوان مثال، «ثانیه» یک واحد زمان است و کاربران از این کلمه برای بیان معیارهای نگهداری یا عملکرد مانند «زمان پاسخ» یا «زمان پاسخ اصلاح» استفاده می‌کنند. کلمه «نصب» نشان می‌دهد که برای عملیات یا نگهداری نرم‌افزار استفاده می‌شود و انتظار داریم با چنین معیاری مواجه شویم.

به‌عنوان یک مثال، دو نیازمندی غیرکارکردی که شامل کلمات «ثانیه» و «نصب» در مجموعه داده هستند در شکل ٦ نشان داده شده‌اند. معمولاً در دنیای واقعی، یک کارشناس

انسانی از این کلمات کلیدی برای دسته‌بندی یک نیازمندی به‌عنوان نیازمندی غیرکارکردی استفاده می‌کند. به‌نظر می‌رسد که مدل این کلمات را به‌عنوان بخشی از فرآیند تصمیم‌گیری خود برای پیش‌بینی یک نیازمندی به‌عنوان نیازمندی غیرکارکردی در نظر می‌گیرد. در مجموعه داده، بیشتر وقوع‌های این کلمه متعلق به نیازمندی‌های غیرکارکردی هستند.

فهرست کلمات حمایتی تولید شده توسط الگوریتم LIME توسط کارشناسان انسانی مورد تجزیه‌وتحلیل قرار گرفت. کارشناسان قصد داشتند مجموعه‌ای از کلمات را استخراج کنند که معمولاً در دنیای واقعی برای دسته‌بندی یک نیازمندی به‌عنوان نیازمندی غیرکارکردی استفاده می‌شوند. جدول ۳ بخشی از این کلمات را که توسط مدل پیشنهادی برای پیش‌بینی یک نیازمندی به‌عنوان نیازمندی غیرکارکردی استفاده می‌شود، نشان می‌دهد.

برخی از کلمات در بین بیش از یک زیرکلاس از نیازمندی‌های غیرکارکردی مشترک هستند. با این حال، هر کلمه تنها در یک زیرکلاس از نیازمندی غیرکارکردی نشان داده می‌شود که در آن بالاترین احتمال وقوع را دارد. این آزمایش نشان می‌دهد که طبقه‌بند تقریباً مشابه یک کارشناس انسانی در تعیین یک نیازمندی به‌عنوان نیازمندی غیرکارکردی عمل می‌کند. در مقایسه با کلمات عمومی مانند «محصول» یا «کاربر»، این کلمات خاص دارای فراوانی کمتری هستند. اما، آن‌ها به‌طور موفقیت‌آمیزی توسط مدل برای پیش‌بینی یک نیازمندی به‌عنوان نیازمندی غیرکارکردی استفاده می‌شوند.

۵-۳- تحلیل قابلیت کاربرد

با در نظر گرفتن نتایج به‌دست آمده از اعمال الگوریتم LIME روی مجموعه داده، می‌توان مشاهده کرد که طبقه‌بند تقریباً از ویژگی‌های مرتبط (مانند کلمات) برای طبقه‌بندی نیازمندی استفاده می‌کند. با این حال، برخی از کلمات عمومی مانند «باید»، «کاربر»، «سیستم» و «محصولات» به‌عنوان

Req 1: When purchasing a streaming movie or pre-paid card via credit card the processing time should have a maximum response time of 15 seconds.
Req 2: The product will function along-side server software on any operating system where the Java runtime can be installed.

.....

نیازمندی یک: هنگام خرید یک فیلم استریم شده یا کارت پیش‌پرداخت از طریق کارت اعتباری، زمان پردازش باید حداکثر زمان پاسخ ۱۵ ثانیه باشد.

نیازمندی دو: این محصول در کنار نرم‌افزار کارساز در هر سیستم‌عاملی که جاوا قابل نصب باشد، کار خواهد کرد.

شکل ۶. نمونه نیازمندی‌هایی که به‌درستی توسط طبقه‌بند به‌عنوان نیازمندی غیرکارکردی طبقه‌بندی شده‌اند.

جدول ۳. زیرمجموعه‌ای از کلمات استفاده‌شده توسط روش پیشنهادی برای پیش‌بینی نیازمندی‌ها به‌عنوان نیازمندی‌های غیرکارکردی

کلمه	کلاس نیازمندی غیرکارکردی
موجود، شکست	در دسترس بودن[1]
عملیات	تحمل خطا[2]
قوانین	قانونی[3]
ناوبری، حل	ظاهر و احساس[4]
بودجه	قابلیت نگهداری[5]
نصب، مجوز	عملیاتی[6]
ثانیه، حداکثر، بار	عملکرد[7]
عملیاتی	قابل حمل[8]
پشتیبانی، هم‌زمان، افزایش	مقیاس‌پذیری[9]
ورود به سیستم، تهدید	امنیت[10]
درک، صفحه، راحتی	قابلیت استفاده[11]

1-Availability
2-Fault Tolerance
3-Legal
4-Look & Feel
5-Maintainability
6-Operational
7-Performance
8-Portability
9-Scalability
10-Security
11-Usability

هم حمایتی و هم مختل‌کننـــده در طبقه‌بندی یک نیازمندی به‌عنوان غیرکارکردی عمل می‌کنند. به‌نظر می‌رسد که این نوع کلمات در ترکیب با سایر کلمات حمایتی، یک نیازمندی را به‌عنوان غیرکارکردی طبقه‌بندی می‌کنند. همچنین، نتایج نشان‌دهنده برخی افعالی هستند که معمولاً برای مشخص کردن قابلیت (یا وظیفه) یک سیستم استفاده می‌شوند. این افعال به مجموعه مختل‌کننـــده تعلق دارند و به‌نظر منطقی می‌رسند.

به‌طور کلی، آزمایش‌ها نشـــان می‌دهنـــد که XAI برای تحلیل نیازمندی و به‌ویژه مشـــکل طبقه‌بنـــدی نیازمندی، کاربردی اســـت. یـــک روش جدید برای اعمـــال XAI روی مشـــکل طبقه‌بندی نیازمندی ارایه شـــد. با این حال، در این کار فقط تحلیل پایه‌ای در نظر گرفته شـــده اســـت و امکان انجام تحلیل‌هـــای دقیق‌تر و پیچیده‌تر وجود دارد. اعضای تیم که مســـئولیت‌هایی در مهندسی نیازمندی دارند، مانند مشـــخص‌کنندگان نیـــاز، تحلیلگران و معمـــاران می‌توانند از توضیح‌دهنده‌هـــای XAI به‌طور هم‌زمـــان با طبقه‌بندها استفاده کنند.

XAI بینش‌هایی برای اعضای تیم درباره پیش‌بینی‌های تولیدشده توســـط مدل‌های یادگیری ماشین فراهم می‌کند. خروجی‌هـــای XAI به آن‌ها کمک می‌کند تـــا بدانند چگونه مدل‌هـــای یادگیری ماشین راه‌حل‌هـــا را تولید می‌کنند. در مورد طبقه‌بندی نیازمندی، XAI به اعضای تیم کمک می‌کند تا کلمات حمایتی، مختل‌کننده و عمومی را در طبقه‌بندی یک نیازمندی به‌عنوان کارکردی، نیازمندی غیرکارکردی یا یک زیرکلاس از نیازمندی غیرکارکردی مشاهده کنند.

۶- نتیجه‌گیری

در ایـــن مقالـــه، قابلیت کاربـــرد XAI در توضیح رفتار الگوریتم‌های طبقه‌بندی نیازمندی مورد مطالعه قرار گرفت. الگوریتم LIME نتایج تولیدشـــده توســـط طبقه‌بند را روی مجموعه داده PROMISE تحلیل کرد. این مطالعه نشان داد که XAI نقش مثبتی در درک بهتر رفتار طبقه‌بندها زمانی که روی نیازمندی‌های متنی اعمال می‌شـــوند، دارد. مهم‌ترین کلماتی که توســـط طبقه‌بندها در دســـته‌بندی نیازمندی‌ها استفاده می‌شوند، شناسایی شـــدند. بررسی و تحلیل این کلمات توسط کارشناسان حوزه (مانند تحلیلگران، معماران و مشـــخص‌کنندگان نیاز) نشـــان داد که کلمات پیشنهادی معنادار هســـتند. در این کار، توضیـــح روش‌های یادگیری ماشیـــن برای طبقه‌بندی نیازمندی مورد توجه قرار گرفت. اخیراً، روش‌های مختلف یادگیـــری عمیق برای طبقه‌بندی نیازمندی‌ها اســـتفاده شده اســـت. توضیح این روش‌ها در حوزه‌های مهندسی نیازمندی ممکن اســـت نتایج جالبی را به‌دلیل عملکرد خوب این روش‌ها در طبقه‌بندی نیازمندی‌ها نشان دهد.

در ســـال‌های اخیر انواع مختلـــفی از توضیح‌دهنده‌ها توســـط محققان طراحی شده‌اند که می‌توانند در این حوزه اســـتفاده شـــوند. در این کار، فقط قابلیـــت کاربرد LIME به‌عنوان یک توضیح‌دهنـــده مورد توجه قرار گرفت. انجام یک مطالعه مقایســـه‌ای بین ایـــن توضیح‌دهنده‌ها توصیه می‌شود. همچنین، توصیه می‌شود که XAI به‌عنوان ابزاری برای انتخاب ویژگی‌ها مورد اســـتفاده قرار گیرد. همچنین یـــکی از چالش‌های الگوریتـــم LIME در زمینـــه پردازش نیازمندی‌های نرم‌افزار این اســـت که توضیحات محلی را با نمونه‌برداری از یک همســـایگی اطـــراف نمونه ورودی ایجاد می‌کند. لذا ایجاد یک همسایگی معنادار و واقع‌گرایانه برای یک نیازمندی نرم‌افزاری می‌تواند تا حدودی دشـــوار باشـــد، به‌ویژه برای نیازمندی‌های خاصی که ممکن است تفسیرهای مختلفی داشـــته باشـــند. برای کاهـــش اثر این محدودیت پیشـــنهاد می‌شود از تکنیک‌های شباهت معنایی برای ایجاد تغییراتی اســـتفاده شـــود که از نظر معنایی به نیازمنـــدی اصلی نزدیک‌تر باشـــند. با ایجـــاد تغییراتی که با زمینه معنایی مطابقت داشـــته باشـــند، می‌توان اطمینان حاصل کرد که همســـایگی مرتبط‌تر و سازگارتر با ماهیت واقعی نیازمندی‌های نرم‌افزاری است.

۷- مراجع

[1] Sommerville, I. (2015). Software engineering. 10th. Book Software Engineering. 10th, Series Software Engineering.
[2] Zhang, Y., Xu, F., Zou, J., Petrosian, O. L., & Krinkin, K. V. (2021, June). XAI evaluation: evaluating black-box model explanations for prediction. In 2021 II International Conference on Neural Networks and Neurotechnologies (NeuroNT) (pp. 13-16). IEEE.
[3] Binkhonain, M., & Zhao, L. (2019). A review of machine learning algorithms for identification and classification of non-functional requirements. Expert Systems with Applications: X, 1, 100001.
[4] Khayashi, F., Jamasb, B., Akbari, R., & Shamsinejadbabaki, P. (2022). Deep Learning Methods for Software Requirement Classification: A Performance Study on the PURE dataset. arXiv preprint arXiv:2211.05286.
[5] Dias Canedo, E., & Cordeiro Mendes, B. (2020). Software requirements classification using machine learning algorithms. Entropy, 22(9), 1057.
[6] Rahimi, N., Eassa, F., & Elrefaei, L. (2021). One-and two-phase software requirement classification using ensemble deep learning. Entropy, 23(10), 1264.
[7] Quba, G. Y., Al Qaisi, H., Althunibat, A., & AlZu'bi, S. (2021, July). Software requirements classification using machine learning algorithm's. In 2021 International Conference on Information Technology (ICIT) (pp. 685-690). IEEE.
[8] Jesus, S., Belém, C., Balayan, V., Bento, J., Saleiro, P., Bizarro, P., & Gama, J. (2021, March). How can I choose an explainer? An application-grounded evaluation of post-hoc explanations. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (pp. 805-815).
[9] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).
[10] Lundberg, S. M., & Lee, S. I. (2017, December). A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 4768-4777).
[11] Habiba, U. E., Bogner, J., & Wagner, S. (2022, August). Can Requirements Engineering Support Explainable Artificial Intelligence? Towards a User-Centric Approach for Explainability Requirements. In 2022 IEEE 30th International Requirements Engineering Conference Workshops (REW) (pp. 162-165). IEEE.
[12] Clement, T., Kemmerzell, N., Abdelaal, M., & Amberg, M. (2023). XAIR: A Systematic Metareview of Explainable AI (XAI) Aligned to the Software Development Process. Machine Learning and Knowledge Extraction, 5(1), 78-108.
[13] Jiarpakdee, J., Tantithamthavorn, C. K., & Grundy, J. (2021, May). Practitioners' perceptions of the goals and visual explanations of defect prediction models. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR) (pp. 432-443). IEEE.
[14] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information fusion, 58, 82-115.
[15] Jesus, S., Belém, C., Balayan, V., Bento, J., Saleiro, P., Bizarro, P., & Gama, J. (2021, March). How can I choose an explainer? An application-grounded evaluation of post-hoc explanations. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (pp. 805-815).
[16] Kim, D., Seo, D., Cho, S., & Kang, P. (2019). Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec. Information Sciences, 477, 15-29.
[17] Jabri, S., Dahbi, A., Gadi, T., & Bassir, A. (2018, April). Ranking of text documents using TF-IDF weighting and association rules mining. In 2018 4th international conference on optimization and applications (ICOA) (pp. 1-6). IEEE.
[18] Ramadhani, D. A., Rochimah, S., & Yuhana, U. L. (2015). Classification of non-functional requirements using semantic-FSKNN based ISO/IEC 9126. TELKOMNIKA (Telecommunication Computing Electronics and Control), 13(4), 1456-1465.
[19] Diwali, A., Saeedi, K., Dashtipour, K., Gogate, M., Cambria, E., & Hussain, A. (2023). Sentiment analysis meets explainable artificial intelligence: A survey on explainable sentiment analysis. IEEE Transactions on Affective Computing.
[20] Mathews, S. M. (2019). Explainable artificial intelligence applications in NLP, biomedical, and malware classification: a literature review. In Intelligent Computing: Proceedings of the 2019 Computing Conference, Volume 2 (pp. 1269-1292). Springer International Publishing.
[21] Zahoor, K., & Bawany, N. Z. (2024). Explainable artificial intelligence approach towards classifying educational android app reviews using deep learning. Interactive Learning Environments, 32(9), 5227-5252.
[22] Zhang, H., & Ogasawara, K. (2023). Grad-CAM-based explainable artificial intelligence related to medical text processing. Bioengineering, 10(9), 1070.
[23] Pospelova, N., Tarasova, A., Subbotina, N., Koroleva, N., Raimova, N., & Lydia, E. L. (2024). Explainable Artificial Intelligence and Natural Language Processing for Unraveling Deceptive Contents. Fusion: Practice & Applications, 14(2).
[24] Chazette, L., Brunotte, W., & Speith, T. (2022). Explainable software systems: from requirements analysis to system evaluation. Requirements Engineering, 27(4), 457-487.
[25] Magableh, A. A. (2023, August). Towards leveraging explainable artificial intelligent (XAI) in requirements engineering (RE) to identify aspect (crosscutting concern): a systematic literature review (SLR) and bibliometric analysis. In 2023 International Conference on Information Technology (ICIT) (pp. 319-326). IEEE.
[26] Bai, S., Shi, S., Han, C., Yang, M., Gupta, B. B., & Arya, V. (2024). Prioritizing user requirements for digital products using explainable artificial intelligence: A data-driven analysis on video conferencing apps. Future Generation Computer Systems, 158, 167-182.