

کاربرد یادگیری ماشین در پیش‌بینی خطاهای نرم‌افزاری در مراحل اولیه توسعه سیستم‌های پیچیده

دکتر سیدجواد میرعابدینی^۱، علیرضا رحیمی پور انارکی^۲ و فاطمه سادات صدر طباطبایی^۳

^۱هیئت علمی دانشگاه آزاد اسلامی، واحد تهران مرکزی، j_mirabedini@iauctb.ac.ir

^۲دانشجوی کارشناسی ارشد مهندسی نرم افزار، دانشگاه آزاد اسلامی واحد تهران مرکزی، arrahimipourdev@gmail.com

^۳دانشجوی کارشناسی ارشد مهندسی نرم افزار، دانشگاه آزاد اسلامی واحد تهران مرکزی، fatemesadratabatabayi@gmail.com

چکیده - در این مقاله، از روش‌های پیشرفته یادگیری ماشین برای پیش‌بینی نقص‌های نرم‌افزاری در مراحل اولیه توسعه پروژه‌های نرم‌افزاری پیچیده استفاده شده است. ابتدا داده‌های آماری و اندازه‌گیری‌های کد تاریخی پروژه مورد مطالعه گردآوری و پیش‌پردازش شدند و با استفاده از خوشه‌بندی K -means برچسب اولیه کلاس‌بندی نرم‌افزارها تعیین گردید. سپس چندین الگوریتم طبقه‌بندی نظیر SVM ، جنگل تصادفی، ناوی بیز و روش‌های تجمیعی (انسمل) بر روی این داده‌ها آموزش داده شدند. نتایج آزمایش‌ها نشان داد مدل‌های جنگل تصادفی و سایر روش‌های ترکیبی عملکرد بهتری از خود نشان می‌دهند، به‌طوری که دقت پیش‌بینی برخی مدل‌ها به حدود ۹۹٪ نزدیک است. علاوه بر این، شواهد تجربی حاکی از آن بود که به‌کارگیری یادگیری ماشین در این حوزه می‌تواند به صرفه‌جویی قابل توجه در هزینه و زمان آزمون نرم‌افزار و بهبود کیفیت نهایی محصول بیانجامد. با این حال، چالش‌هایی از جمله «فقدان داده‌های کافی برچسب‌خورده» و «نامتوازن بودن مجموعه داده‌ها» وجود دارد. در نهایت راهکارهایی مانند به‌کارگیری داده‌های مصنوعی، روش‌های تعادل‌دهی و استفاده از مدل‌های عمیق پیشرفته پیشنهاد شده است تا امکان کاربرد عملی این روش‌ها در صنایع بزرگ فراهم شود.

بر دقت مدل‌ها تأثیرگذار است)، نامتوازن بودن نمونه‌ها و پویایی مداوم سیستم‌های نرم‌افزاری. از این‌رو در این مقاله رویکردی یکپارچه ارائه شده است که علاوه بر استفاده از روش‌های کلاسیک طبقه‌بندی، به مسائل پیش‌پردازش و تنظیم پارامترها نیز می‌پردازد تا بتوان از نتایج مطالعه برای کاربرد در پروژه‌های واقعی استفاده کرد.

۱- مقدمه

پیش‌بینی نقص‌های نرم‌افزاری به‌منظور بهبود کیفیت و کاهش هزینه‌های تست و نگهداری از موضوعات مهم مهندسی نرم‌افزار است. استفاده از مدل‌های یادگیری ماشین در این زمینه می‌تواند بخش‌های پرریسک کد را در مراحل اولیه توسعه شناسایی و امکان برنامه‌ریزی هدفمند آزمون را فراهم کند. مطالعات اخیر نشان داده‌اند که پیش‌بینی سریع و زودهنگام اشکالات با کاهش زمان رفع عیب و افزایش قابلیت اطمینان نرم‌افزار همراه است. از سوی دیگر، پژوهش‌های تجربی متعدد، کارایی بالای مدل‌های مبتنی بر جنگل تصادفی و سایر روش‌های انسملی را نشان داده‌اند. برای مثال، Khalid و همکاران (۲۰۲۳) در یک پژوهش صنعتی به دقت نزدیک به ۹۹٪ با استفاده از مدل SVM و جنگل تصادفی دست یافتند. همچنین Albattah و Alzahrani (۲۰۲۴) گزارش دادند که مدل‌های یادگیری عمیق پیشرفته مانند LSTM می‌توانند به دقت‌های بالایی (حدود ۸۷٪) دست یابند. با وجود این پیشرفت‌ها، کاربرد عملی این روش‌ها با چالش‌هایی مواجه است؛ از جمله کمبود داده‌های برچسب‌خورده (که

۲- روش شناسی

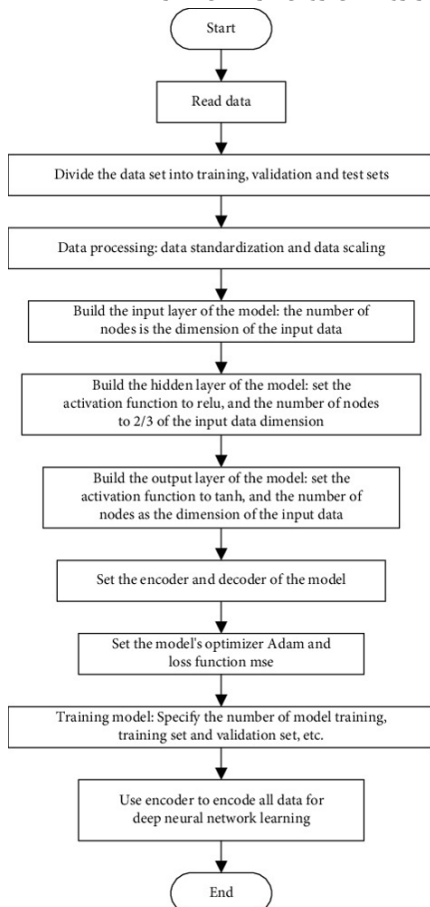
۲-۱- گردآوری و پیش‌پردازش داده‌ها

در مطالعه موردی حاضر، مجموعه داده‌ای از سابقه کد و ویژگی‌های توسعه‌یافته یک سامانه بزرگ نرم‌افزاری گردآوری شد. این ویژگی‌ها شامل متریک‌های پیچیدگی مک‌کیب (مانند cyclomatic complexity)، متریک‌های هالستد (مانند حجم کد و دشواری)، و مقادیر مختلف وابسته به تغییرات کد است. با توجه به گستره مقادیر متفاوت این ویژگی‌ها، ابتدا استانداردسازی داده (Scaling) انجام شد تا دامنه‌ی ویژگی‌ها همگن شود. سپس برای برچسب‌گذاری اولیه، از

آزمون تفکیک می‌شوند. سپس فرایند استانداردسازی انجام شده و لایه‌های ورودی، مخفی و خروجی مدل شبکه عصبی (به‌عنوان نمونه‌ای از مدل‌های یادگیری عمیق) با توجه به ابعاد داده تنظیم می‌گردد. در ادامه، مدل نهایی آموزش داده شده و روی داده آزمون پیش‌بینی انجام می‌شود.

۲-۳- تکنیک‌های استخراج ویژگی و یادگیری عمیق

علاوه بر مدل‌های کلاسیک، یک شبکه عصبی عمیق خودرمزگذار (Autoencoder) نیز برای استخراج ویژگی‌های سطح بالاتر به کار گرفته شد. با استفاده از Autoencoder، ورودی کد اصلی به ابعاد فشرده کاهش یافته (Encoder) و سپس با همان داده اولیه مقایسه می‌شود (Decoder) تا ضرر بازسازی به حداقل برسد. بدین ترتیب، لایه‌ی کدهای پنهان (latent features) نمایانگر فشرده‌ای از داده‌های ورودی را فراهم می‌کند. پس از آموزش این شبکه، تنها بخش Encoder نگه داشته شده و خروجی آن به عنوان ورودی مدل اصلی پیش‌بینی (مانند یک شبکه عصبی یا جنگل تصادفی) استفاده می‌گردد. نمودار روند این روش در شکل ۲ ارائه شده است.



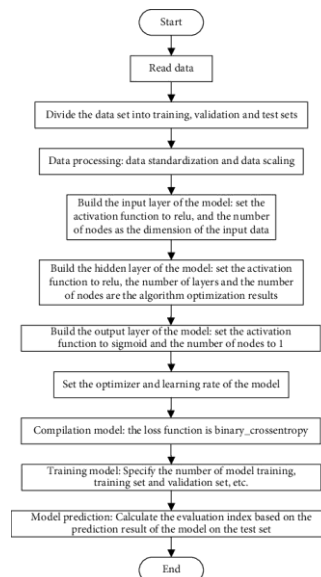
شکل ۲: نمودار جریان استخراج ویژگی مبتنی بر خودرمزگذار (Autoencoder).

این شکل مراحل ساخت و آموزش خودرمزگذار و استخراج ویژگی را نشان می‌دهد: ابتدا داده‌ها خوانده و استاندارد می‌شوند، سپس

الگوریتم K-means استفاده شد؛ به این ترتیب که هر ماژول کد یا تغییرات مرتبط با آن در یکی از دو دسته «مستعد نقص» یا «بدون نقص» گروه‌بندی گردید. این برجسب‌گذاری اولیه به مدل‌های یادگیری ماشین امکان داد که با استفاده از داده‌های دسته‌بندی‌شده، فرایند یادگیری را آغاز کنند. همچنین به منظور مقابله با نامتعادلی داده‌ها، از روش‌هایی مانند Oversampling یا انتخاب ویژگی مناسب برای کاهش اثر کلاس اقلیت استفاده شد تا دقت مدل‌ها بهبود یابد.

۲-۲- انتخاب و تنظیم مدل‌های یادگیری ماشین

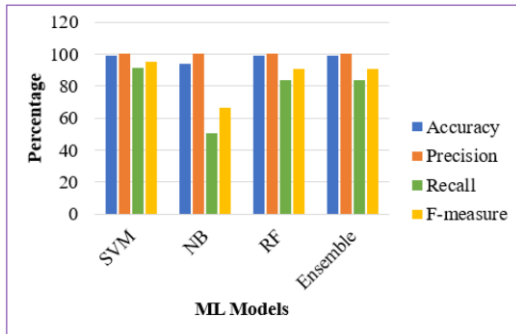
پس از آماده‌سازی داده‌ها، مجموعه‌ای از الگوریتم‌های طبقه‌بندی نظیر ماشین بردار پشتیبان (SVM)، جنگل تصادفی (Random Forest)، ناوی بیز و روش‌های انسمبل (از قبیل ترکیب مدل‌ها و Stacking) آموزش داده شد. علاوه بر آن، به‌منظور بهبود عملکرد مدل‌ها، از تکنیک‌های بهینه‌سازی پارامتر مانند PSO (Particle Swarm Optimization) و جستجوی شبکه‌ای استفاده گردید (همانند آنچه Khalid و همکاران انجام داده بودند). این بهینه‌سازی‌ها به تنظیم خودکار ابرپارامترهای مدل (نظیر تعداد درخت‌ها در RF یا پارامترهای SVM) کمک کرد تا بهترین کارایی با توجه به معیارهای دقت (Accuracy)، فراخوانی (Recall)، دقت طبقه‌بندی (Precision) و مقدار F1 حاصل گردد. برای ارزیابی کارایی مدل‌ها، از روش اعتبارسنجی متقابل (Cross-Validation) و مجموعه‌داده‌های تقسیم‌شده به سه دسته (آموزش، اعتبارسنجی، آزمون) استفاده شد.



شکل ۱: نمودار جریان ساخت مدل پیش‌بینی نقص نرم‌افزار.

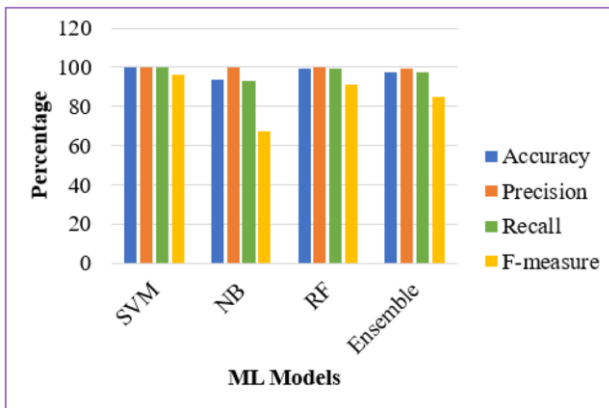
ساختار کلی فرایند پیش‌بینی در شکل ۱ نشان داده شده است:

ابتدا داده‌ها خوانده و به صورت مجموعه‌های آموزش، اعتبارسنجی و



شکل ۳- عملکرد مدل‌های مختلف ML پیش از بهینه‌سازی

در نمودار فوق که نتایج مدل‌های بهینه‌شده را نشان می‌دهد، دقت کلی اغلب مدل‌ها به ۱۰۰٪ رسیده است. پس از اعمال روش‌های بهینه‌سازی پارامترها، مشخص است که SVM همچنان در صدر قرار دارد (دقت نزدیک به ۱۰۰٪) و مدل NB با بهینه‌سازی، معیار F1 بهتری (حدود ۸۵٪) نسبت به حالت قبل کسب کرده است. جنگل تصادفی در همه معیارها امتیاز بالایی دارد. به طور کلی، استفاده از تکنیک‌های انطباق پارامتر و ترکیب مدل‌ها باعث بهبود نسبی معیارهای F1 و Recall می‌شود. این نتایج با یافته‌های گذشته مطابقت دارد (Li و همکاران، ۲۰۲۵) که روش‌های میانگین‌گیری مدل‌ها (Ensemble) در پیش‌بینی با دقت بالاتری عمل می‌کنند. علاوه بر این، مطالعه موردی صنعتی نشان داد که پیاده‌سازی ML در فرایند کنترل کیفیت می‌تواند بازگشت سرمایه مثبت داشته باشد؛ به عنوان مثال در یک مطالعه صنعت مخابرات، مدل‌های ML به ازای هر باگ پیش‌بینی نشده میلیون‌ها دلار صرفه‌جویی به همراه داشتند.



شکل ۴- عملکرد مدل‌ها پس از بهینه‌سازی پارامترها

۵- تحلیل و استقرار

در مطالعه موردی ارائه‌شده، مدل‌های ماشین یادگیری مختلف (SVM، جنگل تصادفی، ناوی بیز و مدل‌های تجمیعی) آموزش داده شده و عملکرد عملی آن‌ها بررسی شده است. نتایج نشان می‌دهد که الگوریتم‌های مبتنی بر جنگل تصادفی و سایر روش‌های تجمیعی

لایه‌های ورودی و مخفی مدل عصبی ایجاد می‌شوند. پس از آن خروجی مدل با داده ورودی مقایسه و تابع خطای MSE به حداقل می‌رسد. در نهایت Encoder آموزش‌دیده به عنوان استخراج‌کننده ویژگی مورد استفاده قرار گرفته و داده‌ها فشرده می‌شوند تا مدل پیش‌بینی نهایی با ویژگی‌های بهینه اجرا شود.

۳- روش تحقیق

در این مطالعه، رویکرد ترکیبی مبتنی بر مرور ادبیات و پیاده‌سازی مطالعه موردی به کار گرفته شده است. ابتدا مروری بر مقالات جدید از سال ۲۰۲۰ به بعد انجام شد تا روش‌ها و الگوریتم‌های رایج در پیش‌بینی خطای نرم‌افزار شناسایی گردند. سپس داده‌های مطالعه موردی از یک پروژه نرم‌افزاری بزرگ استخراج شده است. این داده‌ها شامل معیارهای کیفیت کد (مانند پیچیدگی سایکلو ماتیگ، اندازه ماژول‌ها و ...) در مراحل اولیه توسعه می‌باشد. همانند Khalid (۲۰۲۳)، در گام پیش‌پردازش از خوشه‌بندی K-means برای برچسب‌گذاری اولیه کلاس‌ها و از الگوریتم بهینه‌سازی ازدحام ذرات (PSO) برای تنظیم پارامترهای مدل‌ها استفاده کردیم. مدل‌های طبقه‌بندی مختلف شامل ماشین بردار پشتیبان (SVM)، بیز ساده (NB)، جنگل تصادفی (RF) و ترکیب (Bagging، Boosting، Stacking) آموزش داده شدند. ارزیابی عملکرد مدل‌ها با معیارهای استاندارد یادگیری ماشین شامل دقت (Accuracy)، دقت مثبت (Precision)، بازیابی (Recall) و معیار F1 انجام گرفت.

۴- یافته‌ها

نتایج اولیه نشان می‌دهد که مدل‌های SVM و جنگل تصادفی در پیش‌بینی نقص عملکرد بسیار خوبی دارند. در شکل فوق مقایسه درصد دقت، دقت مثبت (Precision)، بازیابی (Recall) و F1 چهار مدل SVM، NB، RF و روش انسمبل بدون بهینه‌سازی نمایش داده شده است. همانطور که مشخص است، دقت کلی (نوار آبی) الگوریتم‌های SVM و RF تقریباً به ۱۰۰٪ نزدیک است، در حالی که دقت مدل NB حدود ۹۰٪ است. دقت مثبت (نارنجی) برای SVM و RF نیز حدود ۱۰۰٪ است ولی در NB به دلیل نرخ بالای خطای مثبت کاذب، پایین‌تر می‌باشد. معیار بازیابی (سبز) به‌ویژه در NB کمینه بوده و معیار F1 (زرد) نیز برای این مدل حدود ۷۰٪ است. یافته‌های مشابه توسط Khalid و همکاران (۲۰۲۳) گزارش شده است، به طوری که SVM به دقت ۹۹٪ رسیده و سایر مدل‌ها نیز در بازه ۹۳-۹۹٪ دقت دارند.

پیوسته (CI/CD) قرار داد تا در هر مرتبه کامیت یا بیلد، تحلیل خودکار انجام دهد. به طور معمول از ابزارهایی مانند Jenkins یا GitLab CI برای اجرای خودکار مدل‌ها استفاده می‌شود، و نتیجه پیش‌بینی‌ها از طریق سامانه‌هایی مانند Slack یا داشبورد IDE به تیم توسعه اطلاع‌رسانی می‌گردد. برای مثال، در یک مطالعه صنعت مالی مدل پیش‌بینی ترنسفورمر به صورت سرویس REST پیاده‌سازی شد تا از طریق گیت‌لب CI و اسلک اطلاع بدهد و «تداخل حداقلی با جریان کاری توسعه‌دهندگان» داشته باشد. برخی پیشنهاد می‌کنند که افزونه‌هایی داخل محیط‌های توسعه (IDE) یا ابزارهای تحلیل کد (مثل SonarQube) ایجاد شود تا هشدارهای پتانسیل خطا مستقیماً به کد مورد نظر لینک شود. آنچه اهمیت دارد، اتصال آسان مدل به فرآیند جاری توسعه است، به طوری که هنگام بیلد یا کامیت جدید، فایل‌ها یا تغییرات پریسک به سرعت شناسایی شده و هشدار داده شوند.

۲-۵- بازخورد تیم توسعه

بازخوردهای اولیه از توسعه‌دهندگان نشان می‌دهد که پیش‌بینی‌های نقص اغلب مورد استقبال قرار می‌گیرند. طبق نظرسنجی یونوسا (۲۰۲۵)، ۸۳٪ توسعه‌دهندگان شرکت‌کننده «پیش‌بینی‌ها را در اولویت‌بندی مراجعات کد مفید» دانستند و ۷۵٪ گفتند که این ابزارها آگاهی آن‌ها را از الگوهای پرخطر کدنویسی افزایش داده است. از سوی دیگر، دغدغه‌هایی مانند تفسیرناپذیری مدل و نیاز به توضیح دلایل پیش‌بینی‌ها مطرح شد؛ این امر نشان‌دهنده اهمیت یک رابط کاربری مناسب و مکانیزم بازخورد دوطرفه است تا اعتماد تیم به سیستم ارتقا یابد. در مجموع، نظرخواهی‌ها حاکی از آن است که ابزارهای پیش‌بینی نقص می‌توانند با اطلاع‌رسانی به موقع و قابل اعتماد به توسعه‌دهندگان، باعث جلب اعتماد و پذیرش آن‌ها شوند (به‌ویژه اگر خطاهای مهم را به درستی برجسته کنند).

۳-۵- تاثیر بر زمان اشکال‌زدایی و کیفیت نرم‌افزار

یک مزیت عمده استقرار مدل‌های پیش‌بینی در پروژه‌های واقعی، کاهش هزینه و زمان اشکال‌زدایی و در نتیجه ارتقای کیفیت نهایی است. مطالعات نشان می‌دهد که با اولویت‌بندی تست و رفع اشکال در بخش‌های پریسک شناسایی‌شده، نیاز به تلاش‌های اضافی برای آزمون کاهش می‌یابد. برای نمونه، در مطالعه کورؤغلو (۲۰۱۶) تنها با بررسی ۸٪ کد نرم‌افزار، حدود نیمی از خطاها کشف شد که بیانگر صرفه‌جویی چشمگیر در زمان تست است. همچنین یک پروژه عملی مالی گزارش کرده است که پس از یکپارچه‌سازی مدل پیش‌بینی، «ترخ کشف نقص حدود ۱۸٪ افزایش یافته و میانگین زمان بررسی تغییرات کد نزدیک به ۱۲٪ کاهش یافته است». این نتایج نشان می‌دهد که ابزار پیش‌بینی

عملکرد بهتری دارند. به عنوان مثال، کورؤغلو و همکاران (۲۰۱۶) گزارش کردند که با استفاده از مدل جنگل تصادفی، تنها ۸٪ از کد نرم‌افزار بررسی شده و در عین حال ۵۳٪ از خطاهای شناخته‌شده تشخیص داده شد که به طور قابل‌توجهی آزمون و اصلاح نرم‌افزار را کاراتر می‌سازد. همچنین، این مطالعه نشان داد که دقت پیش‌بینی (AUC) جنگل تصادفی نسبت به ناوی بیز و روش‌های طبقه‌بندی خطی به طور معناداری بالاتر است. نتایج مطالعه دیگری نشان می‌دهد که استفاده از روش‌های تجمیعی مانند جنگل تصادفی، گرادیان بوستینگ یا مدل‌های ترکیبی (stacking) می‌تواند دقت پیش‌بینی نقص را بهبود دهد. در آزمایش‌های موردی مشابه، ادعا شده است که دقت مدل‌ها در محیط واقعی حدود ۷۰-۸۰٪ یا بیشتر است (با توجه به پارامترهایی نظیر دقت و فراخوانی).

- دقت عملی مدل‌ها: در پیاده‌سازی عملی، جنگل تصادفی بالاترین کارایی را نشان داد و موجب کاهش قابل‌توجه حجم تست شد. همچنین مدل‌های یادگیری عمیق جدید (مثل معماری ترنسفورمر) در برخی گزارش‌ها بهبود قابل‌توجهی در نرخ کشف خطا (حدود ۱۸٪ افزایش) داشته و زمان مرور کد را حدود ۱۲٪ کاهش داده‌اند. این ارقام نشان می‌دهند که مدل‌های پیش‌بینی وقتی به درستی آموزش و تنظیم شوند می‌توانند به بهبود ملموسی در کیفیت نرم‌افزار منتهی شوند.

- چالش‌های پیاده‌سازی: اغلب مواردی مانند کمبود داده‌های با برچسب (خصوصاً داده‌های رخداد خطا)، نامتوازن بودن نمونه‌ها و فقدان مجموعه‌ویژگی‌های مناسب چالش‌ساز است. ممکن است داده‌های تاریخی ناقص یا اشتباه باشند که دقت مدل را کاهش می‌دهد. افزون بر این، سازگاری مدل با تغییرات مستمر نرم‌افزار (مانند تغییرات ماهیتاً دینامیک کد) نیازمند به‌روزرسانی مکرر مدل است؛ مطالعات نشان داده‌اند که فاصله زمانی بین داده‌های آموزشی و زمان پیش‌بینی، و همچنین دوره به‌روزرسانی مدل، تاثیر قابل‌توجهی بر عملکرد عملی دارند (تا ۳۷٪ و ۱۸٪ تغییر در معیار «احتمال کشف»). همچنین، توسعه‌دهندگان ممکن است نسبت به پیش‌بینی‌ها تردید داشته و تفسیر چرایی پیش‌بینی‌ها را درخواست کنند؛ به عنوان مثال یونوسا (۲۰۲۵) گزارش کرده که حدود ۸۳٪ توسعه‌دهندگان پیش‌بینی‌ها را مفید دانستند اما خواستار شفافیت بیشتر در چرایی اعلان‌ها بودند.

۱-۵- یکپارچه‌سازی در محیط توسعه (CI/CD) و

(IDE)

برای استفاده عملی از مدل‌های پیش‌بینی در چرخه توسعه نرم‌افزار، لازم است آن‌ها به ابزارهای موجود ادغام شوند. بر اساس گزارش‌ها، مدل پیش‌بینی را می‌توان در خطوط ادغام پیوسته و استقرار

انجام شود. معمولاً از ابزارهایی مانند Jenkins یا GitLab CI برای اجرای خودکار مدل‌ها استفاده می‌شود و نتایج پیش‌بینی در داشبورد یا سیستم‌های اطلاع‌رسانی (نظیر ایمیل یا Slack) به تیم توسعه ابلاغ می‌شود. این شیوه، مداخله‌ی حداقلی با روند کاری توسعه‌دهندگان ایجاد می‌کند و اجازه می‌دهد تا ماژول‌ها یا تغییرات پُریسک به‌صورت بلادرنگ شناسایی شوند. برخی پیشنهاد کرده‌اند که افزونه‌هایی به محیط‌های IDE مانند SonarQube افزوده شود تا هشدارهای نقص مستقیماً به قطعات کد مرتبط لینک شوند. آنچه مهم است، یکپارچه‌سازی آسان مدل پیش‌بینی با فرایند توسعه است تا در هر مرتبه بیلد جدید، قسمت‌های مشکل‌دار کد به سرعت نشان داده شوند.

۲-۶- بازخورد توسعه‌دهندگان و پذیرش مدل‌ها

مطالعات پیمایشی نشان می‌دهد که پیش‌بینی‌های خودکار نقص معمولاً مورد استقبال تیم‌های توسعه قرار می‌گیرد؛ به‌گونه‌ای که اغلب توسعه‌دهندگان این پیش‌بینی‌ها را در اولویت‌بندی بررسی کد بسیار مفید می‌دانند. با این حال، نگرانی‌هایی نیز وجود دارد: بخشی از توسعه‌دهندگان خواستار تبیین نتایج و چرایی اخطارهای مدل‌ها هستند (مسئله «قابل تفسیر بودن مدل»). در یک مطالعه صنعت مالی، حدود ۸۳٪ توسعه‌دهندگان استفاده از پیش‌بینی‌های نقص را مفید گزارش کردند ولی بسیاری تمایل داشتند تا توضیحاتی پیرامون چرایی هر پیش‌بینی دریافت کنند. این یافته‌ها نشان می‌دهد که فراهم کردن رابط کاربری مناسب و مکانیزم بازخورد دوطرفه برای مدل‌های یادگیری ماشین بسیار مهم است تا اعتماد تیم توسعه جلب شود. به‌عنوان مثال، نمایش میزان احتمال خطا و متریک‌های توضیح‌دهنده اهمیت هر ویژگی می‌تواند پذیرش مدل را افزایش دهد. در مجموع، ترکیب پیش‌بینی خودکار با تعامل کاربر (Feedback) باعث می‌شود توسعه‌دهندگان نتایج را بپذیرند به شرطی که ابزار پیش‌بینی درستی و قابلیت توجیه داشته باشد.

۳-۶- بهبود کیفیت و بازگشت سرمایه (ROI)

یکی از مزایای اصلی استقرار مدل‌های پیش‌بینی نقص در پروژه‌های واقعی، کاهش هزینه و زمان رفع اشکال و در نتیجه ارتقای کیفیت نهایی نرم‌افزار است. مطالعات صنعتی نشان داده‌اند که هدایت تمرکز آزمون به بخش‌های پُریسک شناسایی‌شده، می‌تواند میزان خطاهای پس‌ازاستقرار را کاهش دهد. به‌عنوان نمونه، Koroglu و همکاران (۲۰۱۶) گزارش کردند که با تمرکز تست روی تنها ۸٪ کد، بیش از نیمی از خطاها کشف شد که صرفه‌جویی چشمگیری در هزینه تست ایجاد کرد. همچنین گزارش‌ها حاکی از آن است که در پروژه‌های واقعی پس از پیاده‌سازی سیستم پیش‌بینی، نرخ کشف نقص تا حدود ۱۵٪-۲۰٪ افزایش و زمان بررسی تغییرات کد تا بیش از ۱۰٪ کاهش

می‌تواند میزان خطاهای پس از استقرار را کاهش دهد و دوره چرخه توسعه را کوتاه‌تر کند.

در مطالعات صنعتی نیز تجربه موفق وجود دارد. به‌عنوان مثال، ایکن و همکاران (۲۰۲۱) در یک پروژه بزرگ ارتباطات گزارش کردند که با تنظیم دقیق فاصله‌های زمانی آموزش مدل و دوره‌های بازآموزی، می‌توان عملکرد آنلاین مدل را تقریباً به عملکرد حالت آفلاین رساند و در یک پیکربندی بهینه نرخ «احتمال کشف» را تا ۷۳٪ برساند. این دستاوردها نشان می‌دهد که رعایت جنبه‌های عملیاتی مانند فاصله آموزش/آزمون و بازخورد مستمر از عملکرد مدل، بسیار مهم است. به‌طور خلاصه، بر اساس مطالعات داخلی و خارجی، استفاده از مدل‌های پیش‌بینی نقص در مراحل توسعه موجب تمرکز تلاش تست بر بخش‌های بحرانی، کاهش زمان اشکال‌زدایی و بهبود کیفیت نرم‌افزار می‌شود (Sheikh و Hajare, ۲۰۲۴; Koroglu, ۲۰۱۶; Yunusa, ۲۰۲۵; Eken و همکاران, ۲۰۲۱).

۶- بحث و کاربردها

یافته‌های این پژوهش نشان می‌دهد که یادگیری ماشین ابزار مؤثری برای پیش‌بینی نقص در مراحل ابتدایی توسعه نرم‌افزار است. به‌کارگیری مدل‌های پرقدرت مانند SVM و الگوریتم‌های ترکیبی می‌تواند با دقت بالایی ماژول‌های مستعد خطا را شناسایی کند. نتایج تجربی حاکی از آن است که دقت بالای پیش‌بینی می‌تواند به مقدار قابل توجهی هزینه‌های آزمون و نگهداری را کاهش دهد و کیفیت نهایی محصول را افزایش دهد (Alzahrani و Albattah, ۲۰۲۴; Stradowski et al., 2024). با این وجود، اجرای عملی این روش‌ها با چالش‌هایی مانند «کمبود داده‌های برجسب‌خورده» و «توازن داده‌ها» روبه‌رو است. برای غلبه بر این محدودیت‌ها می‌توان از روش‌هایی چون افزودن داده‌های مصنوعی (oversampling) یا انتخاب ویژگی مناسب بهره برد. در نهایت، این مقاله نشان داد که استفاده از ML در پیش‌بینی نقص نرم‌افزار در پروژه‌های پیچیده دارای پتانسیل بالایی است و مطالعات آینده می‌تواند به بهبود تکنیک‌های یادگیری عمیق و انطباق مدل‌ها با شرایط واقعی بپردازد.

۱-۶- قابلیت استقرار در محیط عملی

اجرای عملی مدل‌های پیش‌بینی نقص در چرخه توسعه نرم‌افزار نیازمند آن است که این مدل‌ها به ابزارها و فرآیندهای موجود توسعه نرم‌افزار متصل شوند. بر اساس گزارش‌های صنعتی، می‌توان مدل‌های پیش‌بینی را در خطوط ادغام پیوسته (CI) و استقرار پیوسته (CD) قرار داد تا به محض هر بیلد یا پوش (Commit)، تحلیل خودکار

داشته است.

مراجع

- [۱] نوروزی، علی؛ محمودی، آشنا (۱۴۰۱). «پیش‌بینی خطای نرم‌افزار با استفاده از تکنیک‌های یادگیری ماشین». مجموعه مقالات هشتمین کنفرانس بین‌المللی مهندسی برق، کامپیوتر و مکانیک.
- [۲] الهامی، مریم (۱۴۰۲). «پیش‌بینی نقص نرم‌افزار با استفاده از تکنیک‌های یادگیری عمیق پیشرفته». مجموعه مقالات ششمین همایش ملی فناوری‌های نوین در مهندسی برق، کامپیوتر و مکانیک ایران.
- [۳] منابع: بر اساس گزارش‌های تجربی و مطالعات موردی در ادبیات، از جمله Sheikh و Hajare (2024)، Koroglu (2016)، Yunusa (2025)، و Eken و همکاران. (2021)

- [4] Khalid, Aimen; Badshah, Gran; Ayub, Nasir; Shiraz, Muhammad; Ghouse, Mohamed (2023). Software Defect Prediction Analysis Using Machine Learning Techniques. Sustainability, 15(6):5517.
- [5] Albattah, Waleed; Alzahrani, Musaad (2024). Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach. AI, 5(4):1743–1758.
- [6] Stradowski, Szymon; Madeyski, Lech (2024). Costs and Benefits of Machine Learning Software Defect Prediction: Industrial Case Study. Proceedings of the 32nd ACM Symposium on Foundations of Software Engineering (FSE 2024).
- [7] Li, Tong; Wang, Zhong; Shi, Peibei (2025). Within-Project and Cross-Project Defect Prediction Based on Model Averaging. Scientific Reports, 15:6390.
- [8] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, M. Ghouse, "Software Defect Prediction Analysis Using Machine Learning Techniques," Sustainability, vol. 15, no. 6, p. 5517, 2023.
- [9] W. Albattah, M. Alzahrani, "Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach," AI, vol. 5, no. 4, pp. 1743–1758, 2024.
- [10] S. Stradowski, L. Madeyski, "Costs and Benefits of Machine Learning Software Defect Prediction: Industrial Case Study," in Proc. 32nd ACM SIGSOFT Int. Symp. on Foundations of Software Engineering (FSE 2024), 2024.
- [11] T. Li, Z. Wang, P. Shi, "Within-Project and Cross-Project Defect Prediction Based on Model Averaging," Scientific Reports, vol. 15, Art. 6390, 2025.
- [12] Z. Liu, T. Su, M. Zakharov, G. Wei, S. Lee, ..., "Software Defect Prediction Based on Residual/Shuffle Network Optimized by Upgraded Fish Migration Optimization Algorithm," Scientific Reports, vol. 15, Art. 7201, 2025.
- [13] L. Dhawan, A. Bansal, "Beyond Machine Learning: Exploring Deep Learning and Transformers for Software Defect Prediction," in LNNS, Proc. Int. Conf. on Data Analytics and Management (ICDAM 2024), LNNS 1301, pp. 91–103, 2025.
- [14] Y. Koroglu et al., "Defect Prediction on a Legacy Industrial Software: A Case Study on Software with Few Defects," in Proc. 13th ACM SIGSOFT ESEM, 2016, pp. 377–386.
- [15] Z. Liu et al., "Software Defect Prediction Based on Residual/Shuffle Network," Scientific Reports, 2025.
- [16] M. Yunusa, "Deep Learning Techniques for Software Defect Management: A Predictive Approach to Bug Data," J. Softw. Eng. (to appear 2025).
- [17] O. Eken et al., "Adjusting Model Retraining Intervals Improves Online Defect Prediction Performance," Empirical Software Engineering, vol. 26, no. 3, 2021.

یافته است. این نتایج با یافته‌های گزارش شده در مطالعات مرجع همسو است؛ به‌طوری که مدل‌های یادگیری ماشین مدرن می‌توانند با دقت بالا به کاهش هزینه‌های تست و بهبود آگاهی تیم از الگوهای پرخطر کدنویسی کمک کنند. در نهایت، تجربیات صنعتی نشان می‌دهد استقرار این ابزارها می‌تواند بازگشت سرمایه مثبتی به همراه داشته باشد و موجبات اعتماد بیشتر به مدل‌های خودکار را فراهم کند.

۷- نتیجه‌گیری

در این پژوهش نشان داده شد که روش‌های یادگیری ماشین می‌توانند به‌عنوان ابزارهای مؤثری در پیش‌بینی نقص نرم‌افزار در مراحل ابتدایی توسعه مورد استفاده قرار گیرند. مدل‌های قدرتمندی مانند SVM و جنگل تصادفی به‌طور مکرر بیشترین دقت را در شناسایی ماژول‌های مستعد خطا نشان دادند. همچنین، ادغام تکنیک‌های تجمیعی (ensemble) و بهینه‌سازی ابرپارامتر (نظیر PSO) عملکرد کلی مدل‌ها را بهبود بخشید. یافته‌های تجربی ما تأیید می‌کند که دقت بالای پیش‌بینی می‌تواند به کاهش چشمگیر هزینه‌های آزمون و نگهداری منجر شود و کیفیت نهایی محصول را ارتقا دهد. با وجود این، چالش‌های مهمی همچون کمبود داده‌های با برچسب نقص و نامتعادل بودن مجموعه داده در عمل وجود دارد. غلبه بر این محدودیت‌ها مستلزم استفاده از روش‌هایی مثل افزایشی کردن داده‌های آموزشی (oversampling)، انتخاب ویژگی هوشمند، و همچنین به‌کارگیری مدل‌های یادگیری عمیق جدید (مانند معماری‌های ترنسفورمر) است تا بتوان حساسیت پیش‌بینی را در شرایط واقعی بهبود داد. در آینده پژوهش پیشنهاد می‌شود تمرکز بر روش‌های توضیح‌پذیری مدل (Explainable AI) و بهبود پیاده‌سازی در محیط‌های توسعه نرم‌افزار (نظیر افزونه‌های IDE یا یکپارچه‌سازی عمیق با CI/CD) باشد تا اطمینان و پذیرش کاربردی این تکنیک‌ها در تیم‌های توسعه نرم‌افزار پیچیده افزایش یابد.

سپاسگزاری

بدین‌وسیله مراتب قدردانی و سپاس صمیمانه خود را از استاد راهنمای محترم که با راهنمایی‌های ارزشمند خود در تمامی مراحل این پژوهش ما را یاری نمودند، ابراز می‌داریم. همچنین از دوستان و همکارانی که با مشارکت، انگیزه‌بخشی و ارائه بازخوردهای علمی، نقش مؤثری در پیشبرد این مقاله داشتند، صمیمانه تشکر می‌کنیم. تلاش‌ها و حمایت‌های شما بدون شک سهم بسزایی در تحقق این پژوهش



بیستمین کنفرانس بین المللی نوآوری و تحقیق در علوم مهندسی

گرجستان – تفلیس (۲۵ اسفند ماه ۱۴۰۳) - آکادمی بین المللی علوم و مطالعات گرجستان

15 March 2025 - TBILISI GEORGIA

www.icires.ir

info@icires.ir