DOR: https://csj.isi.org.ir/20.1001.1.2538161.1405.11.41.1.4

DOI:https://doi.org/10.22034/csj.2026.251321

تابستان ۱۴۰۵، دوره ۱۱، شماره ۲، پیاپی ۴۱

دریافت مقاله: ۱۴۰۴/۰۸/۲۰

پذیرش مقاله: ۱۴۰۴/۰۹/۱۶

نوع مقاله: پژوهشی

پیش‌بینی نقص نرم‌افزار با استفاده از یادگیری ماشین ترکیبی خودکار و تکامل ویژگی تطبیقی: رویکردی نوین با دقت و توضیح‌پذیری بالا

ابوالفضل دیباجی

دانشجوی دکتری، دانشکده مهندسی کامپیوتر، دانشگاه امیرکبیر (پلی‌تکنیک)، تهران، ایران

پست الکترونیکی: abolfazl.dibaji@aut.ac.ir

احسان ناظرفرد*

استادیار، دانشکده مهندسی کامپیوتر، دانشگاه امیرکبیر (پلی‌تکنیک)، تهران، ایران

پست الکترونیکی: nazerfard@aut.ac.ir

امیر کلباسی

استادیار، دانشکده مهندسی کامپیوتر، دانشگاه امیرکبیر (پلی‌تکنیک)، تهران، ایران

پست الکترونیکی: kalbasi@aut.ac.ir

علیرضا باقری

دانشیار، دانشکده مهندسی کامپیوتر، دانشگاه امیرکبیر (پلی‌تکنیک)، تهران، ایران

پست الکترونیکی: ar_bagheri@aut.ac.ir

چکیده

پیش‌بینی نقص نرم‌افزار یکی از موضوعات مهم در مهندسی نرم‌افزار است که به بهبود کیفیت و کاهش هزینه‌های توسعه کمک می‌کند. در این پژوهش، روشی به‌نام یادگیری ماشین ترکیبی خودکار با تکامل ویژگی تطبیقی معرفی شده که با ترکیب پیش‌پردازش داده‌ها شامل حذف ویژگی‌های بی‌واریانس، پر کردن مقادیر گمشده با میانه و نرمال‌سازی، تکامل پویای ویژگی‌ها، متعادل‌سازی پیشرفته با NearMiss، بهینه‌سازی چند-هدفه با Optuna، و ایجاد مدل گروهی جدید، دقت و توضیح‌پذیری را در پیش‌بینی نقص نرم‌افزار بهبود می‌بخشد. این روش روی مجموعه داده KC1 از مخزن PROMISE ناسا آزمایش شده و با تولید یازده ویژگی تطبیقی شامل نسبت‌های متریک، ترکیب‌های تعاملی، و تبدیل‌های لگاریتمی، و همچنین متعادل‌سازی کلاس‌ها، به صحت ۹۶/۹ درصد، معیار F1 برابر ۹۶/۸ درصد، یادآوری ۹۳/۸ درصد، و AUC برابر ۹۷/۲ درصد دست یافته است. تحلیل توضیح‌پذیری با SHAP TreeExplainer نشان داد که ویژگی‌های تعداد خطوط کد (loc) و معیار t نقش کلیدی در پیش‌بینی‌ها دارند. مقایسه با مدل‌های پایه (مانند رگرسیون لجستیک و جنگل تصادفی) و چندین مطالعه پیشین نشان‌دهنده برتری روش پیشنهادی در صحت و بهبود قابل‌توجه در معیار یادآوری است که برای شناسایی نقص‌های واقعی

* نویسندهٔ مسئول

اهمیت دارد. این روش با ارائه پایداری بالا در اعتبارسنجی ده‌تایی، توضیح‌پذیری قوی، و بهبود معیارهای کلیدی مانند یادآوری، رویکردی جامع بـــرای پیش‌بینی نقص نرم‌افزار ارائه می‌دهد و می‌تواند در کاربردهای صنعتی و پژوهشی مورد استفاده قرار گیرد.

کلید واژه‌ها: پیش‌بینی نقص نرم‌افزار، یادگیری ماشین، تکامل ویژگی، بهینه‌سازی چند هدفه، تحلیل توضیح‌پذیری

۱- مقدمه

پیش‌بینی نقص نرم‌افزار[1] یـــکی از موضوعات مهم در مهندسی نرم‌افزار اســـت که تأثیـــر قابل‌توجهی بر کیفیت، هزینـــه، و زمـــان توســـعه نرم‌افـــزار دارد [۱]. نقص‌های نرم‌افـــزاری می‌توانند به خرابی سیســـتم، کاهش رضایت کاربـــران، و افزایش هزینه‌های نگهداری منجر شـــوند. از این‌رو، توســـعه روش‌های دقیق و کارآمد برای شناسایی زودهنـــگام نقص‌هـــا در فرایند توســـعه نرم‌افـــزار امری ضروری است. در سال‌های اخیر، یادگیری ماشین به‌عنوان ابزاری قدرتمند برای پیش‌بینی نقص نرم‌افزار مورد توجه قرار گرفته اســـت [۲]. با این حـــال، چالش‌هایی نظیر عدم تعادل کلاس‌ها، پیچیدگی ویژگی‌های نرم‌افزاری، و نیاز به توضیح‌پذیـــری نتایج، عملکرد این روش‌ها را محدود کرده است [۳].

روش‌های ســـنتی یادگیری ماشین، از جمله رگرسیون لجستیک، جنگل تصادفی، و ماشین بردار پشتیبان، هرچند در برخی موارد نتایج قابل قبولی ارائه داده‌اند، اما اغلب در برابر داده‌های نامتعـــادل یا ویژگی‌های پیچیده نرم‌افزاری با محدودیت‌هایی مواجه هســـتند [٤]. در مقابل، روش‌های خودکار یادگیری ماشین[2] با بهینه‌سازی خودکار مدل‌ها و انتخاب ویژگی‌ها[3]، عملکـــرد پیش‌بینی را بهبود بخشیده‌اند [٥]، [٦]. با ایـــن وجود، این روش‌ها معمـــولاً فاقد توانایی تولید ویژگی‌های جدید و تطبیقی یا ارائه توضیحات شفاف در مـــورد تصمیم‌گیری‌های مدل هســـتند که این امر برای کاربردهای صنعتی اهمیت دارد.

در این پژوهش، رویکردی نوین به نام یادگیری ماشین ترکیبی خودکار با تکامل ویژگی تطبیقی پیشنهاد شده است که به‌طور خاص برای پیش‌بینی نقص نرم‌افزار طراحی شده است. روش پیشنهادی با ترکیب پیش‌پردازش داده‌ها شامل حذف ویژگی‌های بی‌واریانس، پر کردن مقادیر گمشـــده با میانه و نرمال‌سازی، تکامل پویای ویژگی‌ها، متعادل‌سازی پیشـــرفته داده‌ها، و بهینه‌ســـازی چندهدفه با اســـتفاده از Optuna، به دقت و کارایی قابل‌توجهی دست یافته است. این روش در معیارهای ارزیابی نظیر صحت، یادآوری، معیار F1، و AUC از مدل‌هـــای پایه مانند جنگل تصادفی، شـــبکه عصبی و گرادیان بوســـتینگ عملکرد بهتری نشان داده و در مقایســـه با روش‌های ارائه شـــده در مطالعات پیشین نیز بهبود قابل‌ملاحظه‌ای داشته است. علاوه‌بر این، روش پیشـــنهادی با بهره‌گیری از تحلیل SHAP[4]، توضیح‌پذیری مناســـبی برای تصمیم‌گیری‌های مدل ارائه می‌دهد که این ویژگی آن را به گزینه‌ای مناسب برای کاربردهای عملی در مهندسی نرم‌افزار تبدیل کرده است.

نوآوری‌های این پژوهش عبارتند از: (۱) تولید خودکار ویژگی‌هـــای تطبیـــقی از طریـــق عملیات ریاضی شـــامل نســـبت‌های متریک به تعداد خطوط کد، ترکیب‌های تعاملی ویژگی‌هـــا، و تبدیل‌هـــای لگاریتمی که امـــکان ثبت روابط غیرخـــطی بین متغیرها را فراهم می‌کنـــد، و قرار دادن این ویژگی‌هـــای جدید در کنار ویژگی‌هـــای مجموعه‌داده‌های اصلی، (۲) ترکیب متعادل‌سازی داده‌ها با روش NearMiss (نسخه ۱) و بهینه‌سازی چندهدفه با ابزار Optuna بر اساس معیار ترکیبی F1-AUC، (۳) ساخت مدل ترکیبی وزن‌دار از چهار مدل برتر شناسایی‌شده که پایداری و دقت پیش‌بینی را افزایـــش می‌دهـــد، و (٤) ارائـــه تحلیـــل توضیح‌پذیری جامع با اســـتفاده از SHAP TreeExplainer برای شناسایی ویژگی‌های تأثیرگـــذار بر پیش‌بینی‌ها. ترکیب این تکنیک‌ها در یـــک رویکرد یکپارچه برای اولیـــن بار در این پژوهش صورت گرفته است.

1-Software Defect Prediction
2-Automated Machine Learning
3-Feature Selection
4-SHapley Additive exPlanations

هـدف این مقاله، تشـریح جزئیات روش پیشـنهادی، ارزیابی عملکرد آن در مقایسه با هشت مدل پایه و چندین مطالعه پیشین، و تحلیل توضیح‌پذیری نتایج با استفاده از داده‌های واقعی نرم‌افزاری اسـت. نتایج آزمایش‌ها نشان می‌دهد که روش پیشـنهادی به دقّـت[۵] ۹۶/۹ درصد معیار F1[۶] برابـر ۹۶/۸ درصد، یادآوری[۷] ۹۳/۸ درصد، و AUC[۸] برابر ۹۷/۲ درصد دست یافته که نسبت به مدل‌های پایه و بهترین روش‌های پیشین، در معیارهای صحت و یادآوری بهبـود چند درصدی دارد. ایـن آزمایش‌ها روی مجموعه داده KC1 از مخزن PROMISE ناسـا انجام شـده اسـت که شـامل ۲۱۰۹ ماژول نرم‌افزاری از یـک پروژه واقعی کنترل‌کننـده پرتاب ماهواره به زبان C++ با مجموع حدود ٤۳ هزار خـط کد (میانگین ۲۰/٤ خط بـه‌ازای هر ماژول، دامنه ۱ تا ۲۸۸ خط) اسـت. برای هر مـاژول، ۲۱ ویژگی نرم‌افزاری شامل تعداد خطوط کد، پیچیدگی سیکلوماتیک، معیارهای هالسـتد، و میزان کوپلینگ اسـتخراج شده و با برچسـب معیوب یا غیرمعیوب مشـخص گردیده اسـت. روش پیشـنهادی با ارائه تحلیل‌های بصری و توضیحات قابل‌فهم، به توسـعه‌دهندگان کمک می‌کنـد تا ماژول‌های دارای ریسـک بالای نقص را به‌طور مؤثرتری شناسـایی کنند. در ادامه، در بخش ۲ پیشینه تحقیق بررسی می‌شود، در بخش ۳ روش پیشـنهادی به‌تفصیل تشـریح می‌گردد، در بخش ٤ نتایج ارزیابی و مقایسـه‌ها ارائه می‌شود، و در بخش ۵ نتیجه‌گیری مطرح خواهد شد.

۲- ادبیات پژوهش

پیش‌بیـنی نقص نرم‌افزار به‌عنوان یـکی از حوزه‌های فعـال در مهنـدسی نرم‌افـزار، در سـال‌های اخیر توجه پژوهشگران زیادی را به خود جلب کرده است [۷]، [۸]. این حوزه با هدف شناسـایی زودهنگام نقص‌های نرم‌افزاری برای بهبود کیفیت و کاهش هزینه‌های توسعه، از روش‌های متنوع یادگیری ماشین بهره می‌برد. با این حال، چالش‌هایی نظیر عـدم تعادل کلاس‌ها، پیچیـدگی ویژگی‌ها، و نیاز به شفافیت در تصمیم‌گیری‌های مدل، محدودیت‌هایی را برای روش‌های موجود ایجـاد کرده‌اند. در این بخش، مطالعات معتبر انجام شده بر روی مجموعه داده‌های رایج پیش‌بینی نقص نرم‌افـزار، به‌ویـژه مجموعه‌داده‌های‌های ناسـا و PROMISE که به‌طور گسـترده در تحقیقات مورد استفاده قرار می‌گیرند، مورد بررسی قرار می‌گیرد.

در سـال‌های اخیر، پژوهشـگران به‌طور فزاینده‌ای به سمت استفاده از الگوریتم‌های بهینه‌سازی متاهیوریستیک و مدل‌های یادگیری عمیق برای بهبود دقت پیش‌بینی نقص حرکت کرده‌اند. آرار و آیان [۹] در سـال ۲۰۱۵ رویکردی نوآورانـه مبتنی بر شـبکه عصـبی Cost-Sensitive ارائه دادند که در آن وزن‌های شـبکه توسط الگوریتم Artificial Bee Colony بهینه‌سـازی می‌شـود. این روش با تعریف تابع خطای NECM، امکان تنظیم تعادل بین نرخ تشـخیص صحیح و نرخ هشدار کاذب را فراهم می‌آورد. در ارزیابی بر روی پنج مجموعه‌داده‌های ناسـا با روش اعتبارسنجی متقابل ده‌تایی، این مدل توانایی مدیریت هزینه‌های متفاوت Misclassification را نشـان داد. در مجموعه‌داده‌های KC1، مدل پیشنهادی آنها به AUC برابر ۸۰ درصد، pd برابر ۷۹ درصـد، pf برابر ۳۳ درصـد، Balance برابر ۷۹ درصد و صحت برابر ٦۹ درصد دسـت یافت و در مقایسـه با بیز ساده[۹]، جنگل تصادفی[۱۰]، Immunos، C4.5 و AIRS رتبه برتر را کسب کرد.

در همین راسـتا، باهاورس و همکاران [۱۰] در سـال ۲۰۲۰ با تمرکز بر مسـئله عدم تعـادل کلاس، رویکردی مبتنی بر ترکیب شبکه عصبی و SMOTE [۱۱] پیشنهاد کردند که ویژگی بارز آن بهینه‌سازی هم‌زمان هایپرپارامترهای[۱۲] هر دو بخش (تعداد همسـایه‌ها و نسبت Oversampling در SMOTE و همچنین تعداد لایه‌هـای مخفی، نورون‌ها، نرخ

5-Accuracy
6-F1-Score
7-Recall
8-Area Under Curve

9-Naive Bayes
10-Random Forest
11-Synthetic Minority Oversampling Technique
12-Hyperparameters

یادگیری و dropout در شبکه عصبی) با جستجو تصادفی[13] بـود. این روش بـر روی شـش مجموعه‌داده‌های NASA شامل JM1، KC1، KC2، KC1، PC1، PC3 و PC4 ارزیابی شد. در مجموعه‌داده‌هـای KC1، معیار Balance برابر با ۷۰،۵۴ درصد و یادآوری برابر با ۷۳،۶۲ درصد به‌دست آمد. این نتایج به‌طور معناداری برتر از روش‌های سـنتی یادگیری ماشین همراه با SMOTE قرار گرفت.

انتخاب ویـژگی و کاهش ابعاد نقش محوری در بهبود دقـت و کارایی مدل‌هـای پیش‌بیـنی نقص ایفـا می‌کنند. الخسونه [۱۱] در سـال ۲۰۲۲ با استفاده از شبکه عصبی RBF و انتخاب ویژگی مبتنی بر همبسـتگی، توانست تعداد ویژگی‌هـا را به حـدود ۵۰ درصد کاهـش دهد و در عین حـال دقت پیش‌بینی را حفظ کند. این مدل بر روی چهارده مجموعه‌داده‌هـای NASA بـا روش اعتبارسـنجی متقابل پنج‌تـایی ارزیابی شـد. در مجموعه‌داده‌های KC1، پس از انتخاب ویژگی‌هـا، دقّت برابـر ۸۳،۲۵ درصد، دقت برابر ۸۳،۲۴ درصد، یادآوری برابر ۸۶،۲۴ درصد و F-Measure برابر ۸۳ درصد به‌دست آمد که نسبت به روش‌های پیشین (بیز ساده، MLP، ماشین بردار پشـتیبان[14] و J48) برتری نشان داد.

به‌طور مشـابه، محمود و همکاران [۱۲] در سال ۲۰۲۳ بـا ارزیابی ده الگوریتـم یادگیری ماشین (شـامل جنگل تصـادفی، رگرسیون لجسـتیک[15]، Multilayer Perceptron، بیز سـاده، Lazy IBK، J48، ZeroR، ماشین بردار پشتیبان، شـبکه عصبی[16] و Decision Stump) بر روی ابزار WEKA نشان دادند که اعمال انتخاب ویژگی به‌طور یکنواخت دقّت پیش‌بیـنی را در همـه مجموعه‌داده‌هـای افزایش می‌دهد. در مجموعه‌داده‌هـای KC1، پـس از اعمال انتخاب ویژگی، صحت به ۸۵،۹۱ درصد رسید که بهبود معناداری نسـبت به حالت بدون انتخاب ویژگی نشان داد.

در همین زمینه، مک‌موری و سـودرو [۱۳] در سـال ۲۰۲۳ تحقیـق جامعی انجـام دادند که تأثیـر تکنیک‌های کاهـش ابعـاد (PCA و PLS) و انتخـاب ویـژگی (Fisher score، RFE و Elastic Net) را همـراه با SMOTE بر بهبود عملکـرد یازده مدل یادگیری ماشیـن و یادگیری جمعی[17] بـررسی کرد. ارزیابی بـر روی مجموعه‌داده‌های NASA MDP و PROMISE انجام شـد و یافته‌های آنها نشـان داد کـه PLS به‌دلیل ماهیت نظارت‌شـده و تمرکز بر پیش‌بینی کلاس، بهتریـن و پایدارترین بهبود را ایجـاد می‌کند. در مجموعه‌داده‌هـای KC1، ترکیـب PLS بـا RFE/Fisher و SMOTE به صحت تقریباً ۸۱/۰۷ درصد، دقت حدود ۸۰/۶۱ درصد، یـادآوری تقریباً ۸۱/۶۸ درصد و F1 حدود ۸۱/۰۶ درصد دست یافت.

افزون بر این، رتنانی و همکاران [۱۴] در سـال ۲۰۲۴ بـا اسـتفاده از Ant Colony Optimization بـرای انتخـاب ویـژگی و ترکیـب آن بـا SMOTE و تقویـت گرادیان[18]، نشـان دادند که انتخاب ویژگی‌های بهینه می‌تواند عملکرد مدل‌هـا را در محیط‌های دارای عـدم تعادل کلاس و نوفه به‌طور چشـمگیری بهبود بخشـد. این روش بر روی سه مجموعه‌داده‌های NASA شامل MC1، KC1 و PC2 ارزیابی شـد. در مجموعه‌داده‌هـای KC1، جنگل تصـادفی پس از ترکیـب با ACO و تقویت گرادیان به دقّت برابر ۸۴ درصد، دقـت برابر ۸۵ درصد و یادآوری برابر ۸۲ درصد دسـت یافت که پایدارترین عملکرد را در میان الگوریتم‌های مورد بررسی نشان داد.

رویکردهای ترکیـبی و یادگیری جمعی نیز به‌عنوان راهکارهای مؤثر برای افزایش پایـداری و دقت پیش‌بینی شناخته شده‌اند. شریمانکار و همکاران [۱۵] در سال ۲۰۲۲ با مقایسه گسترده ده الگوریتم یادگیری ماشین (بیز ساده، ماشین بردار پشـتیبان، درخت تصمیم[19]، جنگل تصادفی، آدابوسـت[20]، تقویت گرادیان، رگرسیون لجسـتیک، XG-Boost، MLP و K نزدیکتریـن همسـایه[21]) بر روی دوازده

13-Random Search
14-Support vector machine (SVM)
15-Logistic Regression
16-Neural Network

17-Ensemble Learning
18-Gradient Boosting
19-Decision Tree
20-AdaBoost
21-K-Nearest Neighbors

مجموعه‌داده‌های ناسا، نشان دادند که تقویت گرادیان و رگرسیون لجستیک در اکثر معیارها برتری دارند. این مطالعه بدون اعمال تکنیک‌های پیش‌پردازش خاص مانند تعادل‌سازی کلاس یا انتخاب ویژگی انجام شد. در مجموعه‌داده‌های KC1، تقویت گرادیان به صحت برابر ۷۸ درصد، دقت برابر ۷۶ درصد، یادآوری برابر ۷۸ درصد و معیار F1 برابر ۷٤ درصد دست یافت و در معیار عملکرد کلی میانگین[۲۲] بر سایر طبقه‌بندها برتری معناداری داشت.

در ادامه، باباتونده و همکاران [۱٦] در سال ۲۰۲۳ با استفاده از Meta-Learner داگینگ[۲۳] و ترکیب پیش‌بینی‌های سه طبقه‌بند پایه (بیز ساده، درخت تصمیم و K نزدیکترین همسایه)، توانستند پایداری و دقت مدل‌ها را در مجموعه‌داده‌های نامتعادل به‌طور قابل‌ملاحظه‌ای افزایش دهند. این روش بر روی نه مجموعه‌داده‌های NASA با اعتبارسنجی متقابل ده‌تایی ارزیابی شد. در مجموعه‌داده‌های Dagging_NB ،KC1 به صحت برابر ۸۳/۸ درصد، Dagging_DT به ۸۳/٤٤ درصد و Dagging_kNN به ۸۳/۲ درصد دست یافت که به‌طور معناداری بهتر از طبقه‌بندهای پایه بود. همچنین، بلوئمکه و بورسوکیز [۱۷] در سال ۲۰۲۳ عملکرد جنگل تصادفی (با تعداد درخت ۱۰۰–۱۰۰۰) را با ترکیب SMOTE و بهینه‌سازی هایپرپارامتر بررسی کردند و نشان دادند که این ترکیب بهترین نتایج را ارائه می‌دهد. در مجموعه‌داده‌های KC1، این روش به صحت تقریباً ۹۱ درصد، یادآوری حدود ۹۱ درصد و Balanced تقریباً ۰/۹۱ رسید، در حالی که تنظیم آستانه[۲٤] ناپایدار و غیرقابل‌اعتماد شناخته شد.

صیدیکا و همکاران [۱۸] در سال ۲۰۲۵ با معرفی رویکردی جامع شامل هفده الگوریتم یادگیری ماشین در سه دسته نظارت‌شده، نیمه‌نظارت‌شده و خودنظارت‌شده، و اعمال تکنیک‌های یادگیری جمعی (Boosting، Bagging و Stacking)، تأکید کردند که ترکیب مدل‌های متنوع با یادگیری جمعی به‌طور معناداری دقت و تعادل بین دقت و یادآوری را افزایش می‌دهد. این روش بر روی مجموعه‌داده‌های GHPR (بر پایه KC1، شامل ۲۱۰۹ نمونه با ۲۲ ویژگی) ارزیابی شد و نتایج نشان داد که تقویت گرادیان با صحت برابر ۹۰/۱۰ درصد و آدابوست با ۸۹/٦۰ درصد بهترین عملکرد را داشتند، و مدل‌های یادگیری جمعی (به‌ویژه Boosted رگرسیون لجستیک و Bagged Ridge Classifier) به صحت بیش از ۹۰ درصد، F1 برابر ٦٤ درصد و یادآوری برابر ۸۹ درصد دست یافتند.

استفاده از الگوریتم‌های الهام‌گرفته از طبیعت برای بهینه‌سازی مدل‌های پیش‌بینی نقص، توجه قابل‌توجهی را به خود جلب کرده است. شانکار و چوداری [۱۹] در سال ۲۰۲٤ با استفاده از الگوریتم‌های زیستی مانند الگوریتم خفاش[۲۵]، الگوریتم جستجوی گنجشک[۲٦] و الگوریتم جستجوی سنجاب[۲۷] برای انتخاب ویژگی، نشان دادند که ترکیب این الگوریتم‌ها با شبکه‌های هم‌آمیختی[۲۸] می‌تواند به نتایج برجسته‌ای منجر شود. این روش بر روی چهار مجموعه‌داده‌های PROMISE ارزیابی شد و در مجموعه‌داده‌های KC1، ترکیب الگوریتم خفاش با شبکه هم‌آمیختی به دقّت برابر ۹۱/۲ درصد دست یافت که نسبت به سایر ترکیب‌ها برتری معناداری نشان داد.

همین‌طور، داس و همکاران [۲۰] در سال ۲۰۲۵ دو الگوریتم هیبریدی جدید SSA-WPA (ترکیب الگوریتم جستجوی گنجشک با الگوریتم گروه گرگ[۲۹]) و SSA-ABC (ترکیب الگوریتم جستجوی گنجشک با الگوریتم کلونی زنبور عسل[۳۰]) پیشنهاد کردند که با ترکیب مزایای الگوریتم‌های متاهیوریستیک[۳۱] مختلف، بهینه‌سازی هایپرپارامترهای ANN و XGBoost را به‌طور مؤثر انجام می‌دهند. این روش بر روی پنج مجموعه‌داده‌های NASA ارزیابی شد و در مجموعه‌داده‌های SSA-WPA ،KC1 با ANN به دقّت برابر ۸۸/۷۱ درصد و با XGBoost به ۸۸/۵

22-Overall Performance
23-Dagging
24-Threshold Adjustment

25-BAT Algorithm
26-Sparrow Search
27-Squirrel Search
28-CNN
29-Wolf Pack Algorithm
30-Artificial Bee Colony
31-Metaheuristic

درصد دست یافت؛ همچنین SSA-ABC با ANN به ۸۷/۲۸ درصد رسید. این نتایج نشان داد که ترکیب هوشمند مزایای اکتشاف WPA/ABC با دقت جستجوی SSA، سرعت همگرایی، جلوگیری از بهینه محلی و صحت پیش‌بینی را به‌طور چشمگیری افزایش می‌دهد.

علاوه بر این، عبدو و همکاران [۲۱] در سال ۲۰۲۵، RH-CPDP را برای پیش‌بینی عیوب بین‌پروژه‌ای پیشنهاد کردند که با استفاده از PCA برای کاهش ابعاد و حذف هم‌خطی چندگانه[۳۲] و سپس ترکیب معیارهای استاتیک، ویژگی‌های نحوی (AST + Word2vec) و معنایی گرافی (CFG + DDG + Node2vec) در یک شبکه هیبریدی (hierarchical CNN + MLP با سازوکار Gated Merging) عمل می‌کند. این روش بر روی ۹ پروژه از مجموعه‌داده‌های PROMISE شامل Ant, Jedit, Camel, Log4j, Xalan, Synapse, Lucene, Poi, Xerces ارزیابی شد. نتایج نشان داد RH-CPDP در معیارهای AUC و F1 به‌طور میانگین برتر از روش‌های TPTL، TCSBoost، DBN، DA-KTSVMO و 3SW-MSTL است؛ بهترین عملکرد در جفت Lucene و Ant با AUC برابر ۷/۷۵ درصد و F1 برابر ۷۰/۵ درصد به‌دست آمد که بهبود بسیار خوبی در معیارهای AUC و F1 نسبت به بهترین روش‌های موجود را نشان می‌دهد. این روش به‌دلیل ترکیب هوشمندانه ویژگی‌های معنایی و کاهش مؤثر ابعاد، عملکرد قابل‌توجهی در پیش‌بینی عیوب بین‌پروژه‌ای ارائه کرد.

شبکه‌های بیزی به‌عنوان رویکردهای احتمالی، مزایای منحصربه‌فردی در زمینه تفسیرپذیری[۳۳] و مدیریت عدم قطعیت ارائه می‌دهند. هرناندز- مولینوس و همکاران [۲۲] در سال ۲۰۲۳ سه روش ساخت شبکه بیزی[۳۴] (K2، Hill Climbing و TAN) را ارزیابی کردند و با درخت تصمیم (J48) و جنگل تصادفی مقایسه نمودند. این روش‌ها بر روی سه مجموعه‌داده‌های PROMISE شامل CM1، JM1 و KC1 با روش اعتبارسنجی متقاطع ده‌تایی اجرا شد و نشان دادند که K2 و Hill Climbing کمترین تغییرات (Standard Deviation نزدیک به صفر در بسیاری موارد) و پایداری بالایی دارند. در مجموعه‌داده‌های KC1، بهترین دقّت تقریبا ۸۸/۶۲ درصد به‌دست آمد. این مطالعه تأکید کرد که شبکه‌های بیزی به‌دلیل پایداری بالا و ارائه ساختار گرافیکی روابط علّی بین ویژگی‌ها، شفافیت قابل‌توجهی برای تفسیر مدل فراهم می‌کنند.

کومار و همکاران [۲۳] در سال ۲۰۲۵ با استفاده از شبکه بیزی، Information Gain برای انتخاب ویژگی (سه معیار برتر: CBO، SLOC و WMC از مجموعه CK) و استدلال فازی برای ساخت جدول احتمال شرطی، تأکید کردند که این روش در شرایط عدم قطعیت و با نیاز کمتر به داده‌های حجیم، برتری قابل‌توجهی نسبت به روش‌های یادگیری ماشین پیچیده دارد. این مدل بر روی مجموعه‌داده‌های KC1 ارزیابی گردید و با تقسیم داده به آموزش و آزمون به صحت کلی برابر ۷۷/۹۳ درصد دست یافت؛ همچنین دقت در دسته‌بندی ماژول‌های بدون عیب ۸۷/۱۸ درصد و در ماژول‌های دارای عیب ٦٦/٦٦ درصد بود.

عدم تعادل کلاس یکی از چالش‌های اساسی در پیش‌بینی نقص نرم‌افزار است. بنالا و تانتاتی [۲٤] در سال ۲۰۲۳ تأثیر پنج تکنیک Oversampling (Random Oversampling، SMOTE، ADASYN، Safe-Level SMOTE و SVM-SMOTE) را بر بهبود عملکرد پنج طبقه‌بند یادگیری ماشین (J48، جنگل تصادفی، بیز ساده، آدابوست و Bagging) بررسی کردند و نشان دادند که SVM-SMOTE به‌دلیل تولید نمونه‌های اقلیت در منطقه امن (بین support vectors)، بهترین بهبود را ایجاد می‌کند. ارزیابی بر روی نه مجموعه‌داده‌های NASA شامل CM1، JM1، KC2، KC3، MW1، PC1، MC1، MC2 و PC2 انجام شد و جنگل تصادفی پایدارترین طبقه‌بند شناخته شد.

عالم و مستقیم [۲۵] در سال ۲۰۲۵ با معرفی رویکرد نوینی مبتنی بر یادگیری تقویتی[۳۵] و الگوریتم Q-Learning، نشان دادند که این روش می‌تواند به‌صورت پویا با عدم تعادل کلاس سازگار شود و دقت پیش‌بینی را به‌طور مؤثر

32-Multicollinearity
33-Explanations
34-Bayesian Belief Network
35-Reinforcement Learning

افزایش دهد. ایـن روش ابتدا بـا Random Oversampling داده‌هـای آمـوزشی را متعادل کرده و سـپس عامل RL با دریافـت جایزه بـرای پیش‌بینی صحیـح کلاس اقلیت (عیب‌دار) و جریمه برای خطا، سیاسـت بهینه پیش‌بینی را می‌آموزد. ارزیـابی بر روی مجموعه‌داده‌های NASA-MDP انجام شد و در مجموعه‌داده‌های KC1، دقّت برابر ۹۱ درصد، دقت برابر ۸۹ درصد، یادآوری برابر درصد ۹۳ و F1 برابر ۸۷ درصد به‌دسـت آمد؛ همچنین نرخ تشـخیص عیـب ۹۵ درصد و مصرف منابـع ۹۵/۶۷ درصد بود. این نتایج نسبت به روش‌های سنتی مانند RUSBoost، SMOTE و Cost-Sensitive Learning بهبود چشمگیری نشان داد.

در زمینه سناریوهای cross-project و مدل‌های ترکیبی، کومار و ساکسنا [۲۶] در سال ۲۰۲۴ رویکرد نوآورانه‌ای ارائه دادند کـه در آن پیش‌بینی‌های چندیـن طبقه‌بند پایه (شامل ماشین بردار پشـتیبان، جنگل تصادفی، XGBoost، تقویـت گرادیان، درخت تصمیم، رگرسیون لجسـتیک، K نزدیکترین همسـایه، بیز سـاده گوسی[۳۶] و طبقه‌بند بردار پشـتیبان[۳۷]) با یک شبکه عصبی عمیق ترکیب می‌شود. این روش بر روی چهار مجموعه‌داده‌های PROMISE شـامل KC1، JM1، CM1 و PC1 ارزیابی گردید و نتایج نشـان داد کـه Hybrid Model-3 (ترکیـب KNeighbors + GaussianNB + SVC + Neural Network) در همـه معیارهـا برتری دارد. در مجموعه‌داده‌هـای KC1، این مدل به صحت برابر ۹۵/۳ درصـد، دقت برابـر ۹۵/۹ درصد، یـادآوری برابر ۸۰/۳ درصـد، F1 برابر ۸۴/۱ درصـد و ROC-AUC برابر ۹۳/۹ درصد دسـت یافت که به‌ویژه در شرایط Cross-Project و عدم تعادل کلاس، دقت و تعمیم‌پذیری پیش‌بینی را به‌طور چشمگیری افزایش داد.

مطالعـات مقایسـه‌ای نقـش مهـمی در شناسـایی الگوریتم‌های برتر و درک محدودیت‌های روش‌های موجود ایفا می‌کنند. اقبال و همکاران [۲۷] در سـال ۲۰۱۹ عملکرد ده طبقه‌بند یادگیری ماشین (بیز ساده، RBF، MLP، ماشین بردار پشتیبان، K نزدیکترین همسایه، PART، OneR، kStar، درخـت تصمیم و جنگل تصادفی) را بـدون پیش‌پردازش خاص بـر روی دوازده مجموعه‌داده‌های ناسـا ارزیابی کردنـد. نتایج آنها نشـان داد کـه RBF و جنگل تصادفی پایدارتریـن عملکرد را دارند، اما به‌دلیل عدم تعادل کلاس، بسیاری از مدل‌هـا در یادآوری کلاس اقلیت ضعیف عمل کردنـد. در مجموعه‌داده‌هـای KC1، RBF به صحت برابر ۷۸/۷۹ درصد دست یافت، که اهمیت استفاده از تکنیک‌های متعادل‌سازی را برجسته می‌سازد.

در همین راستا، ساکیب و همکاران [۲۸] در سال ۲۰۲۵ با ارزیابی سیزده الگوریتم یادگیری ماشین نظارت‌شـده (شـامل جنگل تصـادفی، تقویت گرادیـان، ماشین بردار پشتیبان، MLP، رگرسیون لجستیک، K نزدیکترین همسایه، بیز سـاده، درخـت تصمیـم، XGBoost، ANN، LDA، RBF و آدابوسـت) و اسـتفاده از بهینه‌سـازی GridSearchCV و اعتبارسـنجی متقاطـع ده‌تایی، امکان مقایسـه دقیق و جامـع مدل‌ها را فراهم کردند. ایـن مطالعه بر روی چهار مجموعه‌داده‌هـای NASA شـامل KC1، PC1، CM1 و KC2 انجـام شـد. در مجموعه‌داده‌هـای KC1، MLP به صحت برابـر ۸۵/۶۱ درصد، تقویت گرادیان بـه ۸۳/۹ درصد و جنـگل تصادفی به ۸۳/۵ درصد دسـت یافت؛ همچنین در معیارهای دیگر ماننـد FPR پایین (RBF با ۰/۰۰۴) و FNR مناسـب (تقویت گرادیان تقریباً ۰/۶۸) برتری داشت. نتایج نشان داد که جنگل تصادفی و تقویت گرادیان به‌دلیل تعادل عـالی بین دقت و یادآوری، نـرخ خطای پایین و ROC بالا (بالای ۰/۸ در اکثر موارد)، پایدارترین و مؤثرترین مدل‌ها در محیط‌های نامتعادل و پیچیده هستند.

بـررسی جامـع مطالعـات انجام‌شـده بـر روی مجموعه‌داده‌های KC1 و سایر مجموعه داده‌های پیش‌بینی نقص نرم‌افـزار، روندهای مهـمی را در تکامل روش‌های یادگیری ماشین نشـان می‌دهد. جـدول (۱) خلاصه‌ای از مطالعات پیشین را ارائه می‌دهد.

36-Gaussian Naive Bayes (GaussianNB)
37-Support Vector Classifier (SVC)

جدول (۱): خلاصه مطالعات پیشین انجام شده

مرجع	نویسندگان	سال انتشار	تکنیک‌ها	مجموعه‌داده‌های مورد استفاده
[۱۸]	صیدیکا و همکاران	۲۰۲۵	۱۷ الگوریتم ML (نظارت‌شده، نیمه‌نظارت شده، خودنظارت شده), Ensemble	GHPR بر پایه KC1
[۲۳]	کومار و همکاران	۲۰۲۵	BBN, Information Gain, Fuzzy Reasoning	KC1
[۲۸]	ساکیب و همکاران	۲۰۲۵	۱۳ الگوریتم ML نظارت‌شده	CM1, PC1, KC1, KC2
[۲۵]	عالم و مستقیم	۲۰۲۵	Q-Learning, Random Oversampling	CM1, KC1
[۲۰]	داس و همکاران	۲۰۲۵	SSA-WPA, SSA-ABC, ANN, XGBoost	KC1, KC2, KC3, PC1, PC3
[۲۱]	عبدو و همکاران	۲۰۲۵	RH-CPDP: PCA + Hybrid Hierarchical CNN + MLP + Gated Merging	Ant, Jedit, Camel, Log4j, Xalan, Synapse, Lucene, Poi, Xerces
[۱۹]	شانکار و چوداری	۲۰۲۴	الگوریتم‌های Bio-Inspired, CNN	CM1, KC1, KC2, PC1
[۲۶]	کومار و ساکسنا	۲۰۲۴	Hybrid ML, Deep Neural Network	CM1, JM1, KC1, PC1
[۱۴]	رتنانی و همکاران	۲۰۲۴	ACO, SMOTE, GB Ensemble	MC1, KC1, PC2
[۱۶]	باباتونده و همکاران	۲۰۲۳	Dagging Meta-Learner	PC1, PC3, PC4, PC5, CM1, KC1, KC3, MC2, MW1
[۲۴]	بنالا و تانتاتی	۲۰۲۳	پنج تکنیک Oversampling و ML	CM1, JM1, KC2, KC3, MC1, MC2, MW1, PC1, PC2
[۱۷]	بلوئمکه و بورسوکیز	۲۰۲۳	RF, SMOTE, Optimization	CM1, KC1, KC2, PC1, PC2, PC3
[۱۲]	محمود و همکاران	۲۰۲۳	Feature Selection ، ۱۰ الگوریتم ML	CM1, JM1, KC1, KC2, PC1
[۲۲]	هرناندز-مولینوس و همکاران	۲۰۲۳	شبکه‌های بیزی (K2, Hill Climbing, TAN)	CM1, JM1, KC1
[۱۳]	مک‌موری و سودرو	۲۰۲۳	PCA, PLS, RFE, Elastic Net, SMOTE	ناسا MDP و PROMISE
[۱۵]	شریمانکار و همکاران	۲۰۲۲	مقایسه ۱۰ الگوریتم ML	CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5
[۱۱]	الخسونه	۲۰۲۲	Correlation-Based FS, RBF	CM1, JM1, KC1, KC2, KC3, KC4, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5
[۱۰]	باهاورس و همکاران	۲۰۲۰	Neural Network, SMOTE, Random Search	JM1, KC1, KC2, PC1, PC3, PC4
[۲۷]	اقبال و همکاران	۲۰۱۹	مقایسه ۱۰ الگوریتم ML	CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5
[۹]	آرار و آیان	۲۰۱۵	Cost-Sensitive ANN, ABC	KC1, KC2, CM1, PC1, JM1

با بررسی دقیق مطالعات پیشین، می‌توان محدودیت‌های اساسی زیر را شناسایی کرد:

اول، اکثر روش‌های موجود از ویژگی‌های استخراج‌شده به‌صورت دستی یا از پیش تعریف‌شده استفاده می‌کنند و قادر به تولید خودکار ترکیبات جدید ویژگی‌ها نیستند. در مطالعات بررسی‌شده، تنها در چند مورد به انتخاب ویژگی پرداخته شده است، اما هیچ‌کدام به تولید پویای ویژگی‌های مشتق‌شده از طریق عملگرهای ریاضی و منطقی نپرداخته‌اند. این موضوع می‌تواند منجر به از دست رفتن روابط غیرخطی و پیچیده بین ویژگی‌های اولیه شود

دوم، بسیاری از روش‌های یادگیری عمیق و یادگیری جمعی به‌عنوان «جعبه سیاه[۳۸]» عمل می‌کنند و توضیح قابل‌فهمی از نحوه تصمیم‌گیری ارائه نمی‌دهند. تنها در مطالعات چند مورد که از شبکه‌های بیزی استفاده کرده‌اند، ساختار گرافیکی روابط علّی ارائه شده است، اما این

38-Black Box

روش‌ها نمی‌توانند اهمیت نسبی ویژگی‌های تولیدشده و ترکیبات آنها را به‌صورت کمّی و قابل‌فهم برای مهندسان نرم‌افزار توضیح دهند. این موضوع می‌تواند مانع از پذیرش و استفاده عملی این مدل‌ها در محیط‌های صنعتی شود.

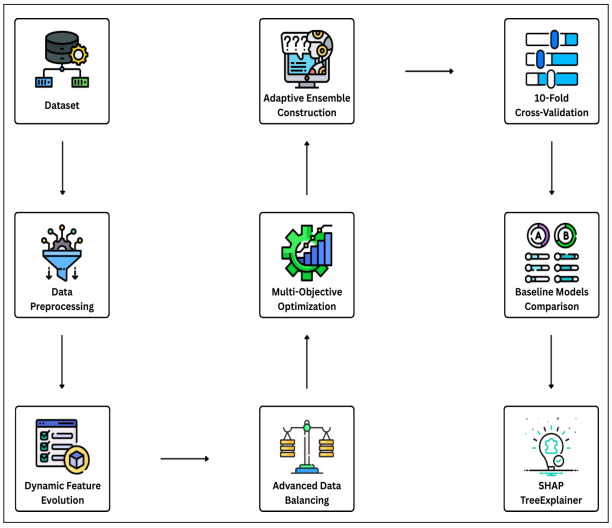
سوم، اکثر مطالعات یا تنها بر انتخاب ویژگی تمرکز کرده‌اند یا تنها بر بهینه‌سازی هایپرپارامترها، و کمتر به بهینه‌سازی هم‌زمان و یکپارچه این دو جنبه پرداخته‌اند.

چهارم، برخی روش‌ها از تکنیک‌های ساده Oversampling مانند SMOTE استفاده کرده‌اند، اما این تکنیک‌ها به‌تنهایی نمی‌توانند به‌طور کامل با پیچیدگی‌های عدم تعادل کلاس در مجموعه‌داده‌های واقعی مقابله کنند. علاوه‌بر این، تکنیک‌های Oversampling داده‌های مصنوعی تولید می‌کنند که این می‌تواند باعث پیش‌بینی غیر واقعی شود. نیاز به رویکردهای ترکیبی و پویا که بتوانند هم‌زمان عدم تعادل کلاس، نوفه و ناهمگنی داده‌ها را مدیریت کنند، به‌وضوح احساس می‌شود.

پنجم، هیچ یک از مطالعات بررسی شده به‌طور جامع و یکپارچه به تمامی این چالش‌ها نپرداخته‌اند. به‌عبارت دیگر، نیاز به رویکردی وجود دارد که بتواند به‌صورت هم‌زمان تولید خودکار ویژگی‌ها، بهینه‌سازی چندهدفه، مقابله مؤثر با عدم تعادل کلاس، و ارائه توضیحات شفاف از تصمیم‌گیری‌های مدل را در یک روش واحد فراهم کند.

۳- مدل پیشنهادی

پیش‌بینی نقص نرم‌افزار یکی از حوزه‌های کلیدی در مهندسی نرم‌افزار است که بهبود کیفیت نرم‌افزار و کاهش هزینه‌های توسعه را هدف قرار داده است. با وجود پیشرفت‌های اخیر در یادگیری ماشین، محدودیت‌هایی نظیر عدم تعادل کلاس‌ها، پیچیدگی ویژگی‌های نرم‌افزاری، و کمبود توضیح‌پذیری در مدل‌های پیش‌بینی، همچنان مانع دستیابی به عملکرد بهینه هستند [۲۹، ۳]. در این راستا، روش پیشنهادی ما، یادگیری ماشین ترکیبی خودکار با تکامل ویژگی تطبیقی، رویکردی جامع ارائه می‌دهد که با تلفیق پیش‌پردازش، تولید ویژگی‌های تطبیقی، متعادل‌سازی پیشرفته داده‌ها، بهینه‌سازی چند-هدفه، و تحلیل توضیح‌پذیری، این چالش‌ها را برطرف می‌کند. این روش با بهره‌گیری از ابزارهای پیشرفته و استراتژی‌های نوین، نه‌تنها دقت پیش‌بینی را بهبود می‌بخشد، بلکه امکان تفسیر نتایج را برای توسعه‌دهندگان فراهم می‌کند. در ادامه، مراحل این روش به‌صورت مفصل تشریح شده و دلایل علمی و منطقی هر مرحله ارائه می‌گردد. شکل ۱ نمودار جریان کاری رویکرد پیشنهادی را نشان می‌دهد.



شکل (۱): نمودار روند کلی روش پیشنهادی

۳-۱ پیش‌پردازش داده‌ها

پیش‌پردازش داده‌ها یکی از مراحل حیاتی در هر سیستم یادگیری ماشین است، زیرا کیفیت داده‌های ورودی مستقیماً بر عملکرد مدل تأثیر می‌گذارد. در روش پیشنهادی، ابتدا ستون ویژگی‌هایی با واریانس صفر حذف می‌شوند، زیرا این ستون‌ها اطلاعات معناداری به مدل اضافه نمی‌کنند و می‌توانند پیچیدگی محاسباتی را افزایش دهند [۳۰]. این اقدام با تحلیل توزیع داده‌ها انجام می‌شود تا اطمینان حاصل شود که هیچ اطلاعات ارزشمندی از دست نمی‌رود. علاوه بر این، برای مدیریت مقادیر گمشده، از راهبرد پر کردن با میانه استفاده می‌شود. میانه به دلیل مقاومت در برابر داده‌های پرت و حفظ توزیع اصلی داده‌ها،

نسبت به میانگین یا روش‌های دیگر برتری دارد [۳۱]. این رویکرد تضمین می‌کند که داده‌ها برای مراحل بعدی تمیز، یکپارچه، و آماده پردازش باشند. این پیش‌پردازش همچنین شامل نرمال‌سازی اولیه داده‌ها برای هماهنگی مقیاس ویژگی‌هاست.

$$X_{norm} = \frac{X - \mu}{\sigma} \quad (۱)$$

با توجه به این که مجموعه داده‌های نرم‌افزاری (مانند KC1) شامل معیارهای متنوعی نظیر تعداد خطوط کد (loc) و پیچیدگی سیکلوماتیک (v(g)) هستند، ناهماهنگی در مقیاس‌ها می‌تواند مدل را به سمت ویژگی‌های با مقادیر بزرگ‌تر منحرف کند. در نتیجه، نرمال‌سازی با روش‌هایی نظیر StandardScaler یا RobustScaler به‌عنوان بخشی از پیش‌پردازش اعمال می‌شود تا این مشکل برطرف گردد. این اقدامات، که با دقت و بر اساس اصول داده‌کاوی انجام می‌شوند، پایه‌ای محکم برای مراحل بعدی فراهم می‌کنند.

$$X_{norm} = \frac{X - Median}{IQR} \quad (۲)$$

۳-۲ تکامل پویای ویژگی‌ها

یکی از نوآوری‌های کلیدی روش پیشنهادی، توانایی تولید و انتخاب ویژگی‌های تطبیقی است که به‌طور خاص برای پیش‌بینی نقص نرم‌افزار طراحی شده‌اند. در این مرحله، ویژگی‌های کلیدی مانند تعداد خطوط کد (loc)، پیچیدگی سیکلوماتیک (complexity)، تعداد عملگرهای یکتا (uniq_op)، و تعداد شاخه‌ها (branchCount) شناسایی می‌شوند. سپس، با استفاده از عملیات ریاضی، ویژگی‌های جدیدی تولید می‌شوند، از جمله:

- نسبت‌های متریک به loc (مانند complexity_per_loc) برای نرمال‌سازی اثر اندازه کد.
- ترکیب‌های تعاملی ویژگی‌ها (مانند v(g)_x_branch-Count) برای ثبت روابط غیرخطی بین متغیرها.
- تبدیل‌های لگاریتمی برای کاهش اثر مقادیر بزرگ و بهبود توزیع ویژگی‌ها.

در مجموع، ۱۱ ویژگی تطبیقی شامل ۳ ویژگی نسبتی، ۳ ویژگی ترکیبی، و ۵ ویژگی لگاریتمی تولید شدند. این ویژگی‌های تطبیقی، به‌منظور غنی‌سازی فضای ویژگی، به‌طور مستقیم به ماتریس ویژگی‌های اولیه الحاق می‌گردند و مجموعه یکپارچه و پویای حاصل، مبنایی غنی و استوار برای مراحل بعدی را فراهم می‌آورد. برای جلوگیری از افزونگی و کاهش ابعاد، ویژگی‌هایی با همبستگی بالاتر از ۰/۰۵ با متغیر هدف (نقص نرم‌افزار) انتخاب می‌شوند.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \quad (۳)$$

این آستانه به‌طور تجربی انتخاب شده تا تعادل بین حفظ اطلاعات مرتبط و حذف ویژگی‌های غیرضروری برقرار شود. در صورتی که تعداد ویژگی‌ها از ۴۰ فراتر رود، تحلیل مؤلفه‌های اصلی[۳۹] اعمال می‌شود تا نفرین ابعاد بالا کاهش یابد [۳۲]. این رویکرد نه‌تنها دقت مدل را با تمرکز بر ویژگی‌های معنادار افزایش می‌دهد، بلکه با تولید ویژگی‌های جدید، محدودیت‌های داده‌های اولیه را برطرف می‌کند و امکان کشف الگوهای پیچیده‌تر را فراهم می‌سازد

۳-۳ متعادل‌سازی پیشرفته داده‌ها

عدم تعادل کلاس‌ها یکی از چالش‌های اصلی در پیش‌بینی نقص نرم‌افزار است، زیرا تعداد نمونه‌های معیوب معمولاً بسیار کمتر از نمونه‌های غیرمعیوب است. برای رفع این مشکل، روش پیشنهادی از روش NearMiss (نسخه ۱) استفاده می‌کند که نمونه‌های اکثریت را بر اساس نزدیکی به نمونه‌های اقلیت در فضای ویژگی‌ها انتخاب می‌کند. این روش نسبت به روش‌های تولید نمونه‌های مصنوعی مانند SMOTE برتری دارد، زیرا با حفظ نمونه‌های واقعی، از ایجاد داده‌های غیرواقعی و خطر بیش‌برازش جلوگیری می‌کند [۳۳]. این اقدام به‌ویژه در مجموعه داده‌هایی مانند

39-Principal Components Analysis

KC1، کـــه تنهـــا ۳۲٦ نمونه معیوب در برابـــر ۱۷۸۳ نمونه غیرمعیوب دارد، اهمیت دارد.

$$d(x_i,x_j) = \sqrt{\sum_{k=1}^{n}(x_{i,k} - x_{j,k})^2} \quad (٤)$$

مجموعه داده به ٦۰ % آموزش ، ۲۰ % اعتبارســـنجی و ۲۰ % آزمـــون تقسیم شـــد و اعتبارســـنجی متقابل ۱۰ برابری روی مجموعه آموزش بـــرای اطمینان از پایداری مدل اعمال شـــد. این تقسیم‌بندی با نســـبت‌های استاندارد انجام می‌شـــود تا ارزیابی مدل‌ها دقیق و بدون ســـوگیری باشد. متعادل‌ســـازی و تقسیم‌بندی داده‌ها تضمین می‌کند که مدل‌ها روی نمونه‌های متنوعی آموزش دیده و توانایی تعمیم‌پذیری بالایی داشـــته باشـــند. این رویکرد، که ریشه در اصول یادگیری ماشین دارد، به بهبود عملکرد مدل در داده‌های نامتعادل کمک می‌کند.

۳-۳ بهینه‌سازی چند-هدفه

برای دســـتیابی به مدل‌های بهینه، روش پیشنهادی از ابزار Optuna با الگوریتم TPE [٤۰] استفاده می‌کند. این ابزار به دلیل توانایی در جستجوی کارآمد فضای هایپرپارامترها، نسبت به روش‌های سنتی مانند GridSearchCV یا جستجو تصادفی برتری دارد [۳٤]. مدل‌های مورد بررسی شـــامل XGBoost،LightGBM و HistGradientBoosting هســـتند که به دلیل ســـرعت بالا، توانایی مدیریـــت داده‌های نامتعادل، و عملکـــرد قـــوی در مســـائل طبقه‌بندی انتخاب شـــده‌اند [۳۵]. علاوه بـــر این، دو مقیاس‌دهنـــده StandardScaler و RobustScaler برای نرمال‌سازی داده‌ها آزمایش می‌شوند تا اثر مقیاس‌های مختلف ویژگی‌ها خنثی شود.

$$Objective = ۰.۶ \cdot F1 - Score + ۰.۴ \cdot AUC \quad (٥)$$

بهینه‌سازی در روش پیشـــنهادی بر اساس یک معیار ترکیـــبی (ترکیـــبی از معیار F1 و AUC) انجام می‌شـــود تا تعادل بین دقت و کارایی محاسباتی حفظ گردد. این معیار ترکیبی امـــکان انتخاب مدل‌هایی را فراهم می‌کند که نه‌تنها دقیق هســـتند، بلکـــه در محیط‌های عملیاتی نیـــز قابل‌اجرا باشـــند. این رویکرد، که با بهره‌گیری از ابزارهای پیشرفته بهینه‌سازی طراحی شـــده، از روش‌های سنتی که اغلب به تنظیم دستی هایپرپارامترها وابسته‌اند، متمایز است.

۳-۵ ساخت یادگیری جمعی (آنسامبل) تطبیقی

به‌منظور افزایش دقت و پایداری، روش پیشنهادی یک یادگیری جمعی تطبیقی از چهار مدل برتر شناسایی‌شـــده توســـط Optuna ایجاد می‌کند. وزن‌های هر مدل بر اساس ترکیبی از معیارهای عملکرد (F1 0.6+ AUC 0.4) تخصیص داده می‌شـــود تا تعادل بین توانایی تشـــخیص کلاس‌های معیوب و غیرمعیوب و دقت کلی حفظ شود. یادگیری تجمعی به دلیل کاهش خطر بیش‌برازش و بهبود تعمیم‌پذیری، در مسائل پیچیده مانند SDP بسیار مؤثر است.

$$P_{ensemble} = \sum_{i=1}^{4} w_i \cdot P_i \quad (٦)$$

این یادگیری جمعی با ترکیـــب پیش‌بینی‌های مدل‌های مختلف (ماننـــد LightGBM و XGBoost) به‌صورت وزن‌دار، عملکردی قوی‌تـــر از هر مدل به‌تنهـــایی ارائه می‌دهد. این رویکرد با اســـتفاده از تنـــوع در مدل‌ها، خطاهای فردی را کاهش داده و پایداری پیش‌بینی‌ها را افزایش می‌دهد. فرایند ســـاخت یادگیری جمعی در روش پیشـــنهادی به‌گونه‌ای طراحی شده که انعطاف‌پذیر بوده و با داده‌ها و معیارهای مختلف سازگار باشد.

۳-۶ ارزیابی با اعتبارسنجی متقاطع

برای اطمینان از تعمیم‌پذیری مدل، اعتبارسنجی متقاطع ده‌تایی در روش پیشـــنهادی اعمال می‌شـــود. این روش با تقسیم داده‌ها بـــه ۱۰ زیرمجموعه، امکان ارزیابی پایدار و

40-Tree-Structured Parzen Estimator

بدون سوگیری را فراهم می‌کند. معیارهای متعددی از جمله صحت، دقت، یادآوری، معیار F1، AUC، ضریب همبستگی متیوز[41]، ویژگی[42]، نرخ خطای مثبت کاذب[43]، خطای میانگین مطلق[44]، ریشه میانگین مربعات خطا[45]، و خطای مربع استاندارد[46] محاسبه می‌شوند. این معیارها به‌طور جامع عملکرد مدل را از جنبه‌های مختلف ارزیابی می‌کنند.

اعتبارسنجی متقاطع ده‌تایی به دلیل توانایی در کاهش تأثیر تغییرات تصادفی در داده‌ها، به‌عنوان استاندارد طلایی در ارزیابی مدل‌های یادگیری ماشین شناخته می‌شود [۳۶]. در رویکرد پیشنهادی، این روش با دقت اجرا می‌شود تا اطمینان حاصل شود که مدل در برابر داده‌های جدید و نادیده عملکرد قابل‌اعتمادی دارد. این مرحله تضمین‌کننده اعتبار علمی روش پیشنهادی است.

برای ارزیابی برتری روش پیشنهادی، عملکرد آن با مدل‌های پایه شامل رگرسیون لجستیک، جنگل تصادفی، ماشین بردار پشتیبان، شبکه عصبی، نزدیک‌ترین همسایه، درخت تصمیم، گرادیان بوستینگ، و آدابوست مقایسه می‌شود. این مدل‌ها به دلیل استفاده گسترده در پیش‌بینی نقص نرم‌افزار انتخاب شده‌اند. مقایسه با استفاده از معیارهای چندگانه انجام می‌شود تا نقاط قوت و ضعف روش پیشنهادی به‌طور دقیق مشخص گردد. این مقایسه نه‌تنها به شناسایی برتری‌های رویکرد پیشنهادی کمک می‌کند، بلکه امکان تحلیل تفاوت‌های آن با روش‌های استاندارد را فراهم می‌سازد. با توجه به این که مدل‌های پایه اغلب فاقد بهینه‌سازی جامع یا توضیح‌پذیری هستند، این مرحله به‌عنوان معیاری برای ارزیابی نوآوری‌های روش پیشنهادی عمل می‌کند.

۳-۷ تحلیل توضیح‌پذیری با SHAP

توضیح‌پذیری یکی از جنبه‌های حیاتی در پیش‌بینی نقص نرم‌افزار است، زیرا توسعه‌دهندگان نیاز دارند دلایل پیش‌بینی‌های مدل را درک کنند. در روش پیشنهادی، از ابزار SHAP با TreeExplainer استفاده می‌شود تا تأثیر هر ویژگی بر پیش‌بینی‌ها به‌صورت کمّی و بصری تحلیل شود. این ابزار با محاسبه مقادیر Shapley، اهمیت ویژگی‌ها را به‌طور دقیق مشخص کرده و امکان شناسایی معیارهای کلیدی (مانند loc یا complexity) را فراهم می‌کند. SHAP TreeExplainer به دلیل سازگاری‌اش با مدل‌های مبتنی بر درخت مانند LightGBM انتخاب شد که امکان محاسبه کارآمد و دقیق اهمیت ویژگی‌ها را فراهم می‌کند.

تحلیل SHAP در روش پیشنهادی برای نمونه‌های معیوب و غیرمعیوب به‌صورت جداگانه انجام می‌شود تا الگوهای متفاوت در هر کلاس شناسایی گردد. این رویکرد توضیح‌پذیری را به سطحی بالاتر از روش‌های سنتی ارتقا می‌دهد، که اغلب به جعبه سیاه محدود می‌شوند.

رویکرد پیشنهادی با تلفیق پیش‌پردازش، تکامل پویای ویژگی‌ها، متعادل‌سازی پیشرفته، بهینه‌سازی چند-هدفه، یادگیری تجمعی، و تحلیل توضیح‌پذیری، رویکردی جامع و نوین برای پیش‌بینی نقص نرم‌افزار ارائه می‌دهد. هر مرحله از این روش با دلایل علمی و منطقی پشتیبانی شده و با بهره‌گیری از ابزارهای پیشرفته مانند Optuna و SHAP، محدودیت‌های روش‌های سنتی را برطرف می‌کند. روش پیشنهادی نه‌تنها برای دستیابی به پیش‌بینی‌های دقیق طراحی شده، بلکه با ارائه توضیحات شفاف، نیازهای عملی توسعه‌دهندگان را نیز برآورده می‌سازد. این روش با انعطاف‌پذیری و جامعیت خود، رویکرد جدیدی در پیش‌بینی نقص نرم‌افزار تعریف می‌کند.

۳-۸ جزئیات پیاده‌سازی

روش پیشنهادی با زبان برنامه‌نویسی Python نسخه ۳,۱۱,۵ پیاده‌سازی شده است. کتابخانه‌های اصلی مورد استفاده شامل Scikit-Learn (نسخه ۱,۵,۱) برای پیاده‌سازی مدل‌های پایه و پیش‌پردازش داده‌ها، Optuna (نسخه ٤,٥,٠) برای بهینه‌سازی چندهدفه هایپرپارامترها، LightGBM (نسخه ٤,٥,٠) و XGBoost (نسخه ۲,۰,۳) برای

41-MCC
42-Specificity
43-FDR
44-MAE
45-RMSE
46-RSE

مدل‌های گرادیان بوستینگ، SHAP (نسخه ۰,۴۷,۲) برای تحلیل توضیح‌پذیری، Imbalanced-Learn (نسخه ۰,۱۳,۰) برای متعادل‌سازی داده‌ها با NearMiss، و NumPy (نسخه ۱,۲۴,۴) و Pandas (نسخه ۲,۰,۳) برای پردازش و مدیریت داده‌ها می‌باشند.

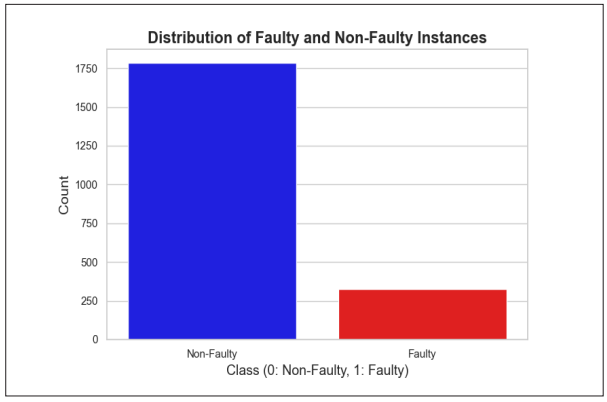
آزمایش‌ها روی سیستمی با پردازنده Intel Core i7 نسل هشتم، ۱۲ گیگابایت حافظه RAM، و کارت گرافیک NVIDIA GeForce GTX 1050 با ٤ گیگابایت حافظه اجرا شدند. برای تضمین تکرارپذیری نتایج، مقدار Seed تصادفی در تمام آزمایش‌ها روی عدد ٤٢ ثابت شده است.

۴- نتایج و بحث

در این بخش، نتایج تجربی روش پیشنهادی یادگیری ماشین ترکیبی خودکار با تکامل ویژگی تطبیقی بر روی مجموعه داده KC1 ارائه و تحلیل می‌شود. این بخش با هدف ارزیابی عملکرد رویکرد پیشنهادی، شناسایی نقاط قوت و محدودیت‌ها، و مقایسه با کارهای پیشین طراحی شده است. نتایج با معیارهای استاندارد یادگیری ماشین شامل صحت، دقت، یادآوری، معیار F1، AUC، ضریب همبستگی متیوز، ویژگی، نرخ خطای مثبت کاذب، خطای میانگین مطلق، ریشه میانگین مربعات خطا، و خطای مربع استاندارد ارزیابی شده‌اند. معیار یادآوری که نسبت نقص‌های صحیح تشخیص‌داده‌شده به کل نقص‌ها را نشان می‌دهد، در پیش‌بینی نقص نرم‌افزار اهمیت ویژه‌ای دارد زیرا عدم شناسایی نقص‌ها می‌تواند هزینه‌های بالایی در مراحل بعدی توسعه ایجاد کند. تحلیل توضیح‌پذیری با SHAP، پایداری مدل با اعتبارسنجی متقاطع ده‌تایی، و مقایسه با مدل‌های پایه و مطالعات پیشین، جامعیت و عملکرد مناسب روش پیشنهادی را نشان می‌دهند. همچنین، پیشنهادهایی برای کارهای آینده ارائه شده تا مسیرهای توسعه این روش مشخص گردد.

۴-۱ تشریح مجموعه داده

مجموعه داده KC1 از مخزن PROMISE ناسا استخراج شده و به یک نرم‌افزار کنترل‌کننده پرتاب ماهواره به زبان C++ مربوط می‌شود [۳۷]، [۳۸]. این مجموعه شامل ۲۱۰۹ ماژول نرم‌افزاری با مجموع ٤٣ هزار خط کد است که اندازه ماژول‌ها از ۱ خط تا ۲۸۸ خط متغیر بوده و میانگین ۲۰,٤ خط به‌ازای هر ماژول است. برای هر ماژول، ۲۱ ویژگی (۲۲ ویژگی با احتساب برچسب هدف) شامل معیارهای کد مانند تعداد خطوط کد (loc)، پیچیدگی سیکلوماتیک (v(g))، تعداد عملگرهای یکتا (uniq_op)، تعداد عملوندهای یکتا (uniq_Opnd)، تعداد کل عملگرها (total_op)، تعداد کل عملوندها (total_Opnd)، تعداد شاخه‌ها (branchCount)، معیارهای هالستد، میزان کوپلینگ، و معیارهای دیگر مانند تعداد خطوط کد منبع (lOCode)، نظرات (lOComment)، و خطوط خالی (lOBlank) استخراج شده است. توزیع کلاس‌ها نامتعادل است، با ۳۲٦ نمونه معیوب (۱۵/٤٦ درصد) و ۱۷۸۳ نمونه غیرمعیوب (٥٤/٨٤ درصد)، که چالشی کلیدی در پیش‌بینی نقص نرم‌افزار محسوب می‌شود. شکل ۲ توزیع کلاس‌های معیوب و غیرمعیوب را به‌صورت بصری نشان می‌دهد و بر عدم تعادل داده‌ها تأکید می‌کند.



شکل (۲): توزیع نمونه‌های معیوب و غیرمعیوب در مجموعه داده KC1

نمونه‌های اولیه KC1 تنوع بالایی در ویژگی‌ها دارند. برای مثال، یک نمونه با loc= 83 ، v(g) =11 و branchCount=11+ به‌عنوان معیوب طبقه‌بندی شده، در

حـــالی که نمونه‌ای با loc برابر ۱/۱ و v(g)= ۱/٤ غیرمعیوب اســـت. این تنوع نشان‌دهنده اهمیت ویژگی‌هایی مانند loc و v(g) در شناسایی نقص‌هاست، که در تحلیل SHAP نیز تأیید شده است. KC1 به دلیل جامعیت، تعداد نمونه‌های کافی، و تنوع ویژگی‌ها، بســـتری ایده‌آل بـــرای ارزیابی روش‌های پیشـــرفته فراهم می‌کند. داده‌ها به ســـه مجموعه آموزش، اعتبارســـنجی، و آزمون تقسیم شـــده‌اند تا ارزیابی بدون ســـوگیری انجام شود. این مجموعه داده به دلیل ارتباط با پروژه‌های واقعی ناســـا، ارزش عمـــلی بالایی در مهندسی نرم‌افزار دارد.

۴-۲ نتایج

نتایج رویکرد پیشنهادی با استفاده از مجموعه داده KC1 و اعتبارسنجی متقاطع ده‌تایی ارزیابی شده‌اند. بهینه‌سازی چند-هدفه با ابزار Optuna بهترین مدل را با الگوریتم Hist-GradientBoosting و پارامترهـــای بهینـــه (max_iter=۱٦۷، max_depth=۳، learning_rate=۰/۱۹۳۳) شناسایی کرد که به امتیاز ترکیبی ۰/۹٦۸۵ (۰/٦ × F1 + ۰/٤ × AUC) رسید. جدول ۲ نتایج اعتبارســـنجی متقاطع ده‌تایی را نشـــان می‌دهد، با میانگین معیار امتیاز F1 برابر ۰/۰۱۷ ± ۰/۹۷۵، که پایداری بالای مدل را تأیید می‌کند. شـــکل ۳ و جـــدول ۲ این نتایج را به‌صورت بصری با بازه اطمینـــان نمایش می‌دهند، که نشان‌دهنده عملکرد یکنواخت در زیرمجموعه‌هاست.

عملکرد نهایی روش پیشـــنهادی روی مجموعه آزمون شامل صحت ۰/۹٦۹، دقت ۱,۰۰۰، یادآوری ۰/۹۳۸، معیار F1 برابـــر ۰/۹٦۸، AUC برابر ۰/۹۷۲، و MCC برابر ۰/۹٤۱ اســـت. معیار یادآوری ۰/۹۳۸ نشـــان می‌دهـــد که روش پیشـــنهادی ۹۳/۸ درصد از نقص‌های واقعی را به‌درستی تشـــخیص می‌دهد، که در کاربردهای عملی اهمیت بالایی دارد زیرا عدم شناســـایی نقص‌هـــا می‌تواند به هزینه‌های قابل‌توجه در مراحل بعدی منجر شود. این معیارها، به‌ویژه AUC بالا (۰/۹۷۲)، نشان‌دهنده تعادل مناسب بین حساسیت و ویژگی است، در حالی که MCC (۰/۹٤۱) همبستگی قوی بین پیش‌بینی‌ها و برچسب‌های واقعی را تأیید می‌کند.

شکل ٤ و جدول ۳ مقایسه معیارهای روش پیشنهادی با مدل‌های پایه مانند رگرسیون لجستیک (صحت ۰/۸٤۱، یادآوری ۰/۱۳۸)، ماشین بردار پشـــتیبان (صحت ۰/۸۵۳، یـــادآوری ۰/۲۰۰)، و جنـــگل تصـــادفی (صحـــت ۰/۸۵۸، یادآوری ۰/۳۸۵) را نشان می‌دهند. روش پیشنهادی به‌طور قابل‌ملاحظه‌ای از تمام مدل‌های پایه عملکرد بهتری داشته اســـت، به‌ویژه در معیار یادآوری که برای پیش‌بینی نقص نرم‌افزار حیاتی است.

تحلیـــل SHAP (همان‌طور که در جدول ٤، شـــکل ۵، و شـــکل ٦ نشان داده شـــده) نشـــان داد که ویژگی‌های loc (۵,۰٤۰۲)، t (۳/۷٦٦۳)، و IOCode (۱/۷۲۷۳) بیشـــترین تأثیـــر را بر پیش‌بینی‌ها دارند. بـــرای نمونه، در نمونه‌های معیوب، مقادیر بـــالای loc (مانند ۲۱/۰۰۰۰ با SHAP برابر ۱/۲۱٤۱-) و t (مانند ۱۲۵/۸۵۰۰ با SHAP برابر ۲/۹٦۲٦) با نقص مرتبط هستند، در حالی که مقادیر پایین در نمونه‌های غیرمعیـــوب (ماننـــد loc برابـــر ٤/۰۰۰۰ بـــا SHAP برابر ۳/٦۵/٤۷-) پیش‌بینی غیرمعیوب را تقویت می‌کنند. شـــکل ۷ روند بهینه‌ســـازی Optuna را نشـــان می‌دهد، که کارایی فرآیند انتخاب پارامترها را تأیید می‌کند.

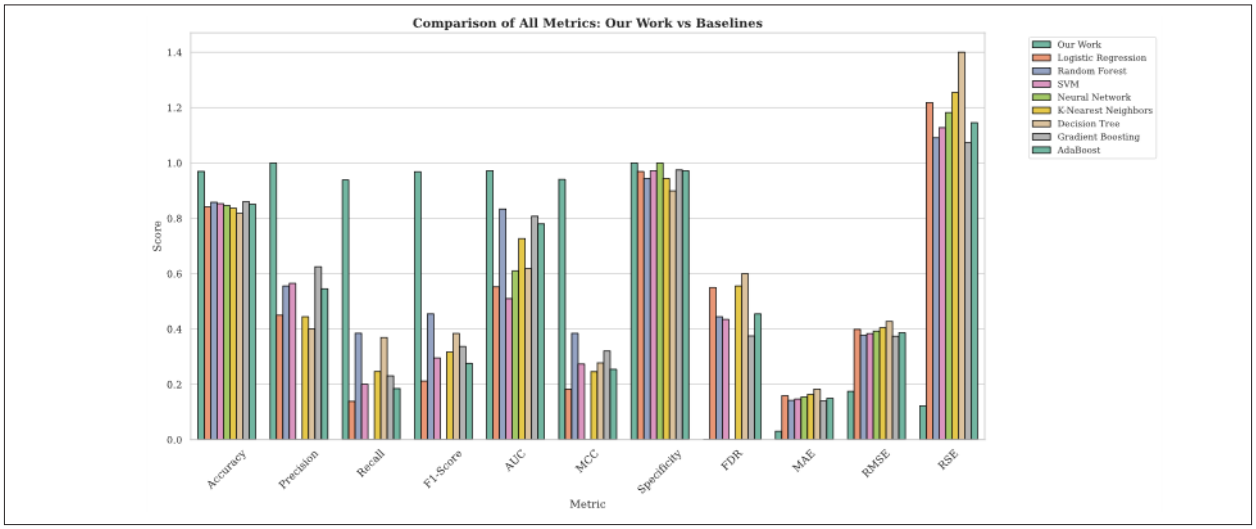
شکل (۳): معیار F1 در اعتبارسنجی متقاطع ده‌تایی

۴-۳ بحث

نتایج روش پیشنهادی نشان‌دهنده عملکرد مناسب این رویکرد در مقایســـه با مدل‌های پایه اســـت. دقّت ۰/۹٦۹، یـــادآوری ۰/۹۳۸، و معیار F1 برابر ۰/۹٦۸، همراه با AUC

جدول (۲): نتایج اعتبارسنجی متقاطع ده‌تایی روش پیشنهادی

Fold	Accuracy	Precision	Recall	F1	AUC	MCC	Specificity	FDR	MAE	RMSE	RSE
۱	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰
۲	۰/۹۵۵	۱/۰۰۰	۰/۹۰۹	۰/۹۵۲	۰/۹۷۳	۰/۹۱۳	۱/۰۰۰	۰/۰۰۰	۰/۰۴۵	۰/۲۱۳	۰/۱۸۲
۳	۰/۹۶۹	۰/۹۴۱	۱/۰۰۰	۰/۹۷۰	۱/۰۰۰	۰/۹۴۰	۰/۹۳۹	۰/۰۵۹	۰/۰۳۱	۰/۱۷۵	۰/۱۲۳
۴	۰/۹۵۴	۱/۰۰۰	۰/۹۰۶	۰/۹۵۱	۰/۹۷۲	۰/۹۱۱	۱/۰۰۰	۰/۰۰۰	۰/۰۴۶	۰/۲۱۵	۰/۱۸۵
۵	۰/۹۶۹	۱/۰۰۰	۰/۹۳۸	۰/۹۶۸	۰/۹۷۹	۰/۹۴۰	۱/۰۰۰	۰/۰۰۰	۰/۰۳۱	۰/۱۷۵	۰/۱۲۳
۶	۰/۹۶۹	۱/۰۰۰	۰/۹۳۸	۰/۹۶۸	۰/۹۷۳	۰/۹۴۰	۱/۰۰۰	۰/۰۰۰	۰/۰۳۱	۰/۱۷۵	۰/۱۲۳
۷	۰/۹۶۹	۱/۰۰۰	۰/۹۳۹	۰/۹۶۹	۰/۹۷۶	۰/۹۴۰	۱/۰۰۰	۰/۰۰۰	۰/۰۳۱	۰/۱۷۵	۰/۱۲۳
۸	۰/۹۸۵	۱/۰۰۰	۰/۹۷۰	۰/۹۸۵	۱/۰۰۰	۰/۹۷۰	۱/۰۰۰	۰/۰۰۰	۰/۰۱۵	۰/۱۲۴	۰/۰۶۲
۹	۰/۹۸۵	۰/۹۷۱	۱/۰۰۰	۰/۹۸۵	۰/۹۹۴	۰/۹۷۰	۰/۹۶۹	۰/۰۲۹	۰/۰۱۵	۰/۱۲۴	۰/۰۶۲
۱۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰

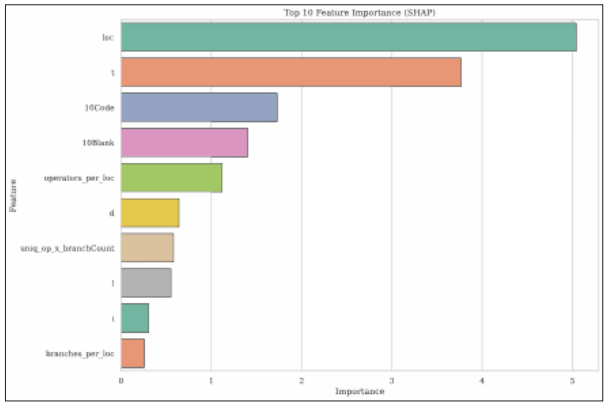


شکل (۴): مقایسه معیارهای عملکرد روش پیشنهادی با مدل‌های پایه

جدول (۳): مقایسه عملکرد روش پیشنهادی با مدل‌های پایه

Model	Accuracy	Precision	Recall	F1	AUC	MCC	Specificity	FDR	MAE	RMSE	RSE
Our Work	۰/۹۶۹	۱/۰۰۰	۰/۹۳۸	۰/۹۶۸	۰/۹۷۲	۰/۹۴۱	۱/۰۰۰	۰/۰۰۰	۰/۰۳۱	۰/۱۷۵	۰/۱۲۲
Gradient Boosting	۸۶۰/۰	۶۲۵/۰	۲۳۱/۰	۳۳۷/۰	۸۰۸/۰	۳۲۰/۰	۹۷۵/۰	۳۷۵/۰	۱۴۰/۰	۳۷۴/۰	۰۷۳/۱
Random Forest	۸۵۸/۰	۵۵۶/۰	۳۸۵/۰	۴۵۵/۰	۸۳۳/۰	۳۸۴/۰	۹۴۴/۰	۴۴۴/۰	۱۴۲/۰	۳۷۷/۰	۰۹۱/۱
SVM	۸۵۳/۰	۵۶۵/۰	۲۰۰/۰	۲۹۵/۰	۵۱۰/۰	۲۷۳/۰	۹۷۲/۰	۴۳۵/۰	۱۴۷/۰	۳۸۳/۰	۱۲۸/۱
AdaBoost	۸۵۱/۰	۵۴۵/۰	۱۸۵/۰	۲۷۶/۰	۷۸۰/۰	۲۵۴/۰	۹۷۲/۰	۴۵۵/۰	۱۴۹/۰	۳۸۶/۰	۱۴۶/۱
Neural Network	۸۴۶/۰	–	–	–	۶۱۰/۰	–	۰۰۰/۱	–	۱۵۴/۰	۳۹۲/۰	۱۸۲/۱
Logistic Regression	۸۴۱/۰	۴۵۰/۰	۱۳۸/۰	۲۱۲/۰	۵۵۲/۰	۱۸۳/۰	۹۶۹/۰	۵۵۰/۰	۱۵۹/۰	۳۹۸/۰	۲۱۸/۱
K-Nearest Neighbors	۸۶۳/۰	۴۴۴/۰	۲۴۶/۰	۳۱۷/۰	۷۲۸۷/۰	۲۴۶/۰	۹۴۴/۰	۵۵۶/۰	۱۶۴/۰	۴۰۴/۰	۲۵۵/۱
Decision Tree	۸۱۸/۰	۴۰۰/۰	۳۶۹/۰	۳۸۴/۰	۶۱۸/۰	۲۷۷/۰	۸۹۹/۰	۶۰۰/۰	۱۸۲/۰	۴۲۷/۰	۴۰۰/۱

برابر ۰/۹۷۲ و MCC با مقدار ۰/۹۴۱، نشان می‌دهد که روش پیشنهادی در شناسایی نمونه‌های معیوب عملکرد خوبی دارد و تعادل قابل‌قبولی بین تشخیص کلاس‌های معیوب و غیرمعیوب برقرار می‌کند. این عملکرد به دلیل ترکیب پیش‌پردازش داده‌ها شامل حذف ویژگی‌های بی‌واریانس، پر کردن مقادیر گمشده با میانه و نرمال‌سازی، تولید



شکل (۵): نمودار میله‌ای اهمیت ویژگی‌ها بر اساس SHAP (۱۰ مورد برتر)

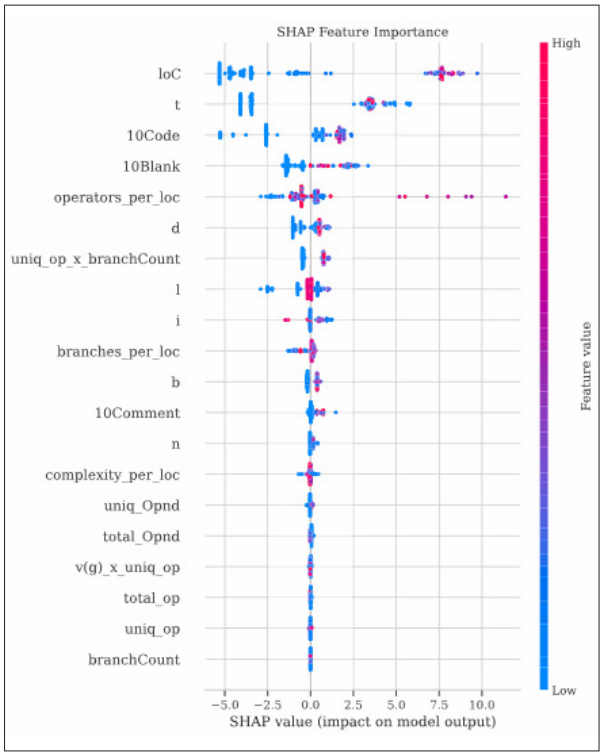
جدول (۴): اهمیت ویژگی‌ها بر اساس SHAP (۱۰مورد برتر)

ویژگی	اهمیت SHAP
loc	۵/۰۴۰۲
t	۳/۷۶۶۳
IOCode	۱/۷۲۷۳
IOBlank	۱/۴۰۴۵
operators_per_loc	۱/۱۲۰۶
d	۰/۶۴۰۱
uniq_op_x_branchCount	۰/۵۷۸۸
l	۰/۵۵۰۹
i	۰/۳۰۶۸
branches_per_loc	۰/۲۵۵۳

شکل (۷): روند بهینه‌سازی Optuna

ویژگی‌های تطبیقی، متعادل‌سازی با NearMiss (نسخه ۱)، و بهینه‌سازی چندهدفه با Optuna است.

بـــرای مثال، تولید ویژگی‌های جدیـــد مانند operators_per_loc و uniq_op_x_branchCount امـــکان ثبت تعاملات



شکل (۶): نمودار خلاصه‌ی تحلیل توضیح‌پذیری SHAP

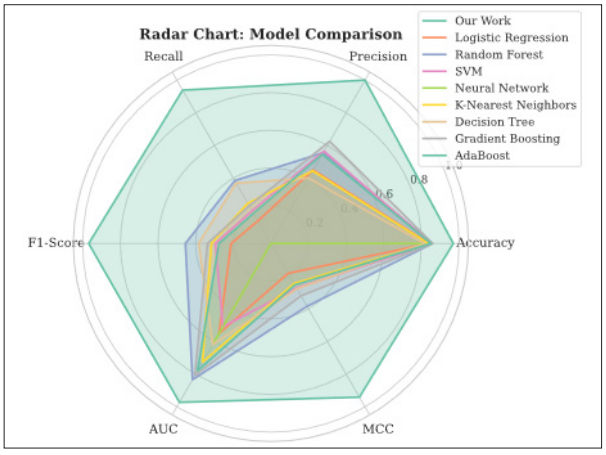
غیرخـــطی بین متغیرها را فراهم کرده، که در مدل‌های پایه مانند رگرسیون لجســـتیک یا ماشین بردار پشتیبان دیده نمی‌شود. همچنین، استفاده از NearMiss (نسخه ۱) به‌جای روش‌های تولیـــد نمونه‌های مصنـــوعی مانند SMOTE، با انتخـــاب نمونه‌های واقعی کلاس اکثریـــت، از بیش‌برازش جلوگیـــری کـــرده و داده‌های متعادل‌شـــده‌ای بـــا کیفیت مناسب‌تر فراهم کرده است.

معیار یادآوری ۰/۹۳۸ نشان می‌دهد که روش پیشنهادی ۹۳/۸درصـــد از نقص‌های واقعی را تشـــخیص می‌دهد که برای کاربردهای عملی در پیش‌بینی نقص نرم‌افزار اهمیت بالایی دارد. در مقایسه، بهترین مدل پایه (تقویت گرادیان) تنها یادآوری ۰/۲۳۱ (۲۳/۱ درصد) داشت که نشان می‌دهد اکثر نقص‌ها را تشـــخیص نمی‌داد. این تفاوت قابل‌توجه در یادآوری، یکی از نقاط قوت اصلی روش پیشنهادی است.

با این حال، روش پیشـــنهادی محدودیت‌هایی نیز دارد. پیچیدگی محاسباتی ناشی از بهینه‌سازی چندهدفه و تحلیل SHAP ممکن است چالش‌برانگیز باشد. علاوه بر این، روش پیشـــنهادی روی مجموعه داده KC1 آزمایش شده و نیاز

به ارزیابی روی مجموعه داده‌هـــای متنوع‌تر (مانند CM1، JM1، یـــا PC1 از مخزن PROMISE، یـــا مجموعه داده‌های از زبان‌های برنامه‌نویسی و حوزه‌های کاربردی مختلف) دارد تا تعمیم‌پذیری آن تأیید شود.

تحلیل SHAP نشـــان داد که ویژگی‌هـــای loc و t نقش مهمی دارند، اما وابســـتگی به این ویژگی‌ها ممکن است در پروژه‌هـــای نرم‌افزاری با معیارهـــای متفاوت یا زبان‌های برنامه‌نویسی دیگر کاهش یابد. شکل ۸ مقایسه چندمعیاره روش پیشـــنهادی با مدل‌هـــای پایه را نشـــان می‌دهد، که عملکرد مناســـب آن در معیارهای مختلف را تأیید می‌کند. این تحلیل‌ها نشان می‌دهند که روش پیشنهادی رویکردی جامع و قابل‌اعتماد برای پیش‌بینی نقص نرم‌افزار است، اما نیازمند ارزیابی‌های بیشتر و بهینه‌سازی برای کاربردهای گسترده‌تر است.



شکل (۸): نمودار راداری مقایسه چندمعیاره روش پیشنهادی با مدل‌های پایه

۴-۴ مقایسه با کارهای پیشین

برای ارزیـــابی جایـــگاه روش پیشـــنهادی در ادبیات پیش‌بینی نقص نرم‌افزار، عملکرد آن با چندین مطالعه اخیر روی مجموعه داده KC1 مقایسه شده است. این مقایسه بر اســـاس دو معیار کلیدی صحت و یادآوری انجام شده که هر دو برای ارزیابی جامع عملکرد مدل‌های پیش‌بینی نقص اهمیت دارند. جدول ۵ خلاصه‌ای از صحت و یادآوری این مطالعات را نشان می‌دهد.

جدول (۵): مقایسه روش پیشنهادی با کارهای پیشین

مرجع	نویسندگان	صحت	یادآوری
[۱۸]	صیدیکا و همکاران	۰/۹۰۱	۰/۸۹
[۲۳]	کومار و همکاران	۰/۷۷۹۳	–
[۲۸]	ساکیب و همکاران	۰/۸۵۶۱	–
[۲۵]	عالم و مستقیم	۰/۹۱	۰/۹۳
[۲۰]	داس و همکاران	۰/۸۸۷۱	–
[۱۹]	شانکار و چوداری	۰/۹۱۲	–
[۲۶]	کومار و ساکسنا	۰/۹۵۳	۰/۸۰۳
[۱۴]	رتنانی و همکاران	۰/۸۴	۰/۸۲
[۱۶]	باباتونده و همکاران	۰/۸۳۸	–
[۱۷]	بلوئمکه و بورسوکیز	۰/۹۱	۰/۹۱
[۱۲]	محمود و همکاران	۰/۸۵۹۱	–
[۲۲]	هرناندز-مولینوس و همکاران	۰/۸۸۶۲	–
[۱۳]	مک‌موری و سودرو	۰/۸۱۰۷	۰,۸۱۶۸
[۱۵]	شریمانکار و همکاران	۰/۷۸	۰,۷۸
[۱۱]	الخسونه	۰/۸۳۲۵	۰,۸۶۲۴
[۱۰]	باهاورس و همکاران	–	۰,۷۳۶۲
[۲۷]	اقبال و همکاران	۰/۷۸۷۹	–
[۹]	آرار و آیان	۰/۶۹	–
		۰/۹۶۹	۰/۹۳۸

روش پیشنهادی

در معیـــار صحت، روش پیشـــنهادی با مقـــدار ۰/۹۶۹ (۶۹/۹ درصد) نســـبت بـــه تمام مطالعـــات پیشین عملکرد بهتری داشته است. نزدیک‌ترین رقیب در این معیار، مطالعه کومار و ساکســـنا [۲۶] با صحت ۰/۹۵۳ اســـت که از مدل ترکیبی شـــامل K نزدیکترین همســـایه، بیز ساده گوسی، طبقه بند بردار پشتیبان و شبکه عصبی استفاده کرده بود. با این حال، این مطالعـــه در معیار یادآوری تنها به ۰/۸۰۳ دست یافت که نشان می‌دهد حدود ۲۰ درصد از نقص‌های

واقعی را تشـــخیص نداده است، که برای کاربردهای عملی قابل‌قبول نیست. این تفاوت نشـــان می‌دهد که صحت بالا به‌تنهایی معیار کافی برای ارزیابی مدل‌های پیش‌بینی نقص نیست و تعادل بین صحت و یادآوری ضروری است. روش پیشنهادی ما با ترکیب متعادل‌سازی پیشرفته NearMiss و بهینه‌سازی چندهدفه، توانسته تعادل بهتری بین صحت و یادآوری برقرار کند.

مطالعه شانکار و چوداری [۱۹] با استفاده از الگوریتم خفـــاش برای انتخاب ویژگی و شـــبکه هم‌آمیختی به دقّت ۰/۹۱۲ رسیـــد. ایـــن روش به‌رغم اســـتفاده از تکنیک‌های پیشـــرفته یادگیری عمیق، فاقد متعادل‌ســـازی مؤثر برای رفـــع عدم تعـــادل کلاس‌ها بـــود و تحلیـــل توضیح‌پذیری ارائـــه نکرد. علاوه بر این، بهینه‌ســـازی هم‌زمان ویژگی‌ها و هایپرپارامترهـــا در آن مطالعه انجام نشـــد، در حالی که روش پیشـــنهادی با استفاده از Optuna این بهینه‌سازی را به‌صورت یکپارچه انجام می‌دهد.

عالم و مستقیم [۲۵] با معرفی رویکرد مبتنی بر یادگیری تقویـــتی (Q-Learning) به صحت ۰/۹۱ دســـت یافتند. این روش با Random Oversampling داده‌ها را متعادل کرده و از جایزه/ جریمه برای بهبود پیش‌بینی کلاس اقلیت استفاده کـــرد. با این حال، تولید داده‌هـــای مصنوعی در این روش می‌تواند منجر به بیش‌بـــرازش و پیش‌بینی‌های غیرواقعی شود، در حالی که روش پیشنهادی با استفاده از NearMiss (نســـخه ۱) که نمونه‌های واقعی را انتخـــاب می‌کند، از این مشکل جلوگیری کرده است.

بلوئمکه و بورســـوکیز [۱۷] با ترکیب جنگل تصادفی و SMOTE به صحت ۰/۹۱ رسیدند. این مطالعه نشان داد که SMOTE می‌توانـــد عملکرد را بهبود بخشـــد، اما محدود به یـــک الگوریتم پایه (جنگل تصادفی) بود و از بهینه‌ســـازی چندهدفه یا تولید ویژگی‌های تطبیقی استفاده نکرد.

در مطالعـــهٔ صیدیکا و همـــکاران [۱۸] با اســـتفاده از تکنیک‌هـــای یادگیری جمعی شـــامل Boosting، Bagging و Stacking به دقّت ۰/۹۰۱ دســـت یافتند. این روش با ترکیب هفده الگوریتم در سه دسته نظارت‌شده، نیمه‌نظارت‌شده و خودنظارت‌شده، تنوع بالایی داشت، اما به تولید ویژگی‌های جدیـــد نپرداخـــت و از ویژگی‌های اولیه اســـتفاده کرد که می‌تواند روابط غیرخطی پیچیده را نادیده بگیرد.

معیـــار یـــادآوری کـــه نســـبت نقص‌هـــای صحیـــح تشخیص‌داده‌شـــده به کل نقص‌ها را نشـــان می‌دهد، برای پیش‌بینی نقص نرم‌افزار از اهمیت بالایی برخوردار اســـت زیرا عدم شناسایی نقص‌ها می‌تواند به هزینه‌های قابل‌توجه در مراحل بعدی توسعه منجر شـــود. روش پیشنهادی با یادآوری ۰/۹۳۸ (۹۳/۸ درصد) نشـــان می‌دهد که تنها ٦/٢ درصد از نقص‌های واقعی را تشخیص نداده است.

بهتریـــن عملکـــرد در معیـــار یـــادآوری در مطالعات پیشین مربوط به عالم و مســـتقیم [۲۵] با مقدار ۰/۹۳ است که بسیار نزدیک به روش پیشـــنهادی اســـت. با این حال، روش آن‌ها با صحت ۰/۹۱ نسبت به صحت ۰/۹٦۹ روش پیشـــنهادی، تفاوت قابل‌توجهی دارد. این نشان می‌دهد که روش پیشـــنهادی توانسته تعادل بهتری بین صحت کلی و شناسایی نقص‌ها برقرار کند.

بلوئمکه و بورســـوکیز [۱۷] با یـــادآوری ۰/۹۱ و دقّت ۰/۹۱، عملکرد متعادلی داشتند، اما همچنان در هر دو معیار کمتر از روش پیشـــنهادی بودند. صیدیکا و همکاران [۱۸] با یادآوری ۰/۸۹ و الخســـونه [۱۱] با یـــادآوری ۰/٨٦٢٤ نیـــز عملکرد قابل‌قبولی داشـــتند، اما در مقایســـه با روش پیشنهادی، درصد قابل‌توجهی از نقص‌ها را از دست دادند

رتنانی و همکاران [١٤] با یادآوری ۰/۸۲، مک‌موری و سودرو [۱۳] با ۰/۸۱٦۸، کومار و ساکسنا [٢٦] با ۰/۸۰۳، آرار و آیـــان [۹] با ۰/۷۹، شـــریمانکار و همکاران [۱۵] با ۰/۷۸، و باهاورس و همکاران [۱۰] با ۰/۷۳٦۲ در این معیار عملکرد ضعیف‌تری نسبت به روش پیشنهادی داشتند. این نشان می‌دهد که بسیاری از مطالعات پیشین در شناسایی نقص‌های واقعی با چالش مواجه بوده‌اند.

عملکرد مناســـب روش پیشـــنهادی در هـــر دو معیار صحت و یادآوری به دلایل زیر است:

اول، تولید خودکار ویژگی‌های تطبیقی: بررسی مطالعات پیشین نشان می‌دهد که اکثر آن‌ها از ویژگی‌های استخراج‌شده به‌صورت دستی یا از پیش تعریف‌شده استفاده کرده‌اند. تنها در مطالعاتی مانند الخسونه [۱۱]، محمود و همکاران [۱۲]، و رتنانی و همکاران [۱٤] به انتخاب ویژگی پرداخته شده، اما هیچ‌کدام به تولید پویای ویژگی‌های مشتق‌شده از طریق عملیات ریاضی نپرداخته‌اند. روش پیشنهادی با تولید ۱۱ ویژگی تطبیقی شامل نسبت‌های متریک (مانند operators_per_loc)، ترکیب‌های تعاملی (مانند uniq_op_x_branchCount)، و تبدیل‌های لگاریتمی، امکان ثبت روابط غیرخطی و پیچیده بین متغیرها را فراهم کرده که در مدل‌های پایه مانند رگرسیون لجستیک یا ماشین بردار پشتیبان دیده نمی‌شود

دوم، متعادل‌سازی مؤثر با حفظ نمونه‌های واقعی: بسیاری از مطالعات پیشین از تکنیک‌های Oversampling مانند SMOTE استفاده کرده‌اند. SMOTE با تولید نمونه‌های مصنوعی در فضای ویژگی، می‌تواند منجر به بیش‌برازش و پیش‌بینی‌های غیرواقعی شود، به‌ویژه در مناطق نویزی یا پرت. روش پیشنهادی با استفاده از NearMiss (نسخه ۱)، نمونه‌های واقعی کلاس اکثریت را بر اساس نزدیکی به نمونه‌های اقلیت انتخاب می‌کند، که از بیش‌برازش جلوگیری کرده و داده‌های متعادل‌شده‌ای با کیفیت بالاتر فراهم می‌کند.

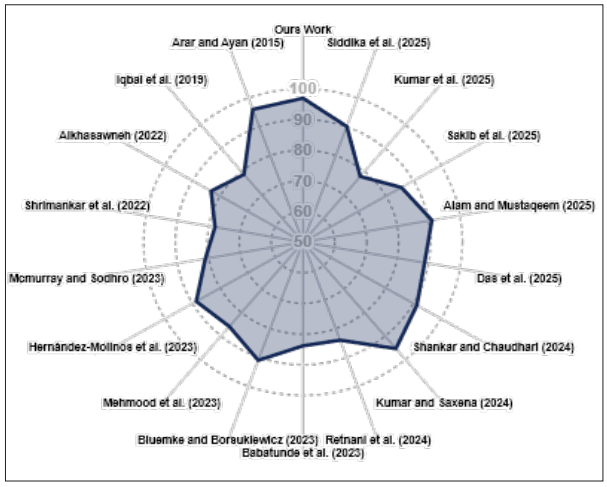
سوم، بهینه‌سازی همزمان ویژگی‌ها و هایپرپارامترها: اکثر مطالعات یا تنها بر انتخاب ویژگی تمرکز کرده‌اند یا تنها بر بهینه‌سازی هایپرپارامترها. روش پیشنهادی با استفاده از Optuna و الگوریتم TPE، بهینه‌سازی چندهدفه را بر اساس معیار ترکیبی (۰/٤ × AUC +۰/٦ × F1) انجام می‌دهد که امکان یافتن تعادل بهینه بین دقت پیش‌بینی و کارایی محاسباتی را فراهم می‌کند. این رویکرد نسبت به روش‌های سنتی مانند GridSearchCV که در مطالعه ساکیب و همکاران [۲۸] استفاده شده، کارآمدتر است.

چهارم، مدل ترکیبی وزن‌دار با تنوع بالا: برخی مطالعات از تکنیک‌های یادگیری جمعی استفاده کرده‌اند، اما اکثر آنها بر ترکیب ساده یا رای‌گیری متمرکز بودند. روش پیشنهادی با ساخت مدل ترکیبی وزن‌دار از چهار مدل برتر شناسایی‌شده توسط Optuna و تخصیص وزن‌ها بر اساس عملکرد، تنوع در پیش‌بینی‌ها را افزایش داده و خطاهای فردی را کاهش می‌دهد.

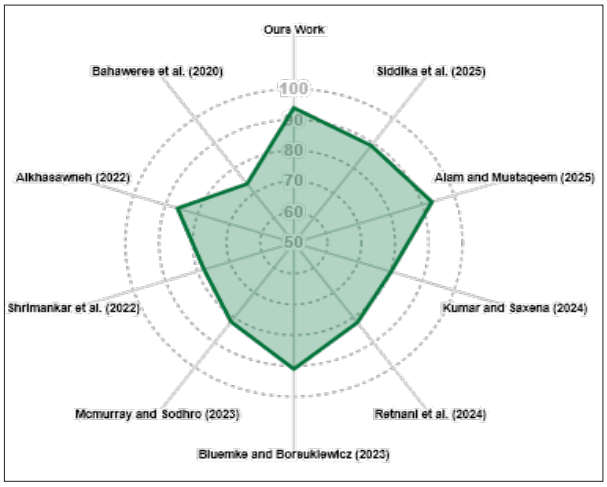
پنجم، توضیح‌پذیری با تحلیل SHAP: یکی از محدودیت‌های اساسی مطالعات پیشین، عدم ارائه توضیحات شفاف از تصمیم‌گیری‌های مدل است. بسیاری از روش‌های یادگیری عمیق و یادگیری جمعی به‌عنوان جعبه سیاه عمل می‌کنند. تنها در مطالعاتی که از شبکه‌های بیزی استفاده کردند، ساختار گرافیکی روابط علّی ارائه شده، اما این روش‌ها نمی‌توانند اهمیت نسبی ویژگی‌های تولیدشده را به‌صورت کمّی توضیح دهند. روش پیشنهادی با استفاده از SHAP TreeExplainer، نه‌تنها اهمیت هر ویژگی را به‌صورت کمّی مشخص می‌کند (مانند loc با اهمیت ۵/۰٤ و t با ۳/۷۷)، بلکه با ارائه نمودارهای بصری مانند summary plot و waterfall plot، درک نتایج را برای توسعه‌دهندگان و مدیران پروژه آسان‌تر می‌کند.

بررسی جامع مطالعات پیشین، محدودیت‌های مشترک زیر را نشان می‌دهد:

- عدم تولید خودکار ویژگی‌ها: هیچ‌یک از مطالعات بررسی‌شده به تولید پویای ویژگی‌های مشتق‌شده نپرداخته‌اند که می‌تواند منجر به از دست رفتن روابط غیرخطی و پیچیده بین ویژگی‌های اولیه شود.
- نبود توضیح‌پذیری کمّی: اکثر روش‌های پیشرفته (به‌ویژه یادگیری عمیق و یادگیری جمعی) فاقد توضیح‌پذیری قابل‌فهم برای مهندسان نرم‌افزار هستند که می‌تواند مانع از پذیرش و استفاده عملی آنها در محیط‌های صنعتی شود.
- بهینه‌سازی جزئی: کمتر مطالعه‌ای به بهینه‌سازی هم‌زمان و یکپارچه انتخاب ویژگی‌ها و هایپرپارامترها پرداخته است.
- وابستگی به داده‌های مصنوعی: استفاده گسترده از



شکل (۹): نمودار راداری مقایسه معیار صحت روش پیشنهادی با کارهای پیشین



شکل (۱۰): نمودار راداری مقایسه معیار یادآوری روش پیشنهادی با کارهای پیشین

SMOTE و روش‌های مشابه می‌تواند منجر به بیش‌برازش و پیش‌بینی‌های غیرواقعی شود.

- عدم جامعیت: هیچ یک از مطالعات بررسی‌شده به‌طور جامع و یکپارچه به تمامی چالش‌های تولید ویژگی، بهینه‌سازی، متعادل‌سازی، و توضیح‌پذیری نپرداخته‌اند.

روش پیشنهادی با دستیابی به دقّت ۰/۹۶۹ و یادآوری ۰/۹۳۸، در مقایسه با چندین مطالعه پیشین، تعادل مناسبی بین دقت کلی و شناسایی نقص‌ها برقرار کرده است. این عملکرد به دلیل ترکیب منحصربه‌فرد پیش‌پردازش داده‌ها، تولید خودکار ویژگی‌های تطبیقی، متعادل‌سازی مؤثر با NearMiss، بهینه‌سازی چندهدفه با Optuna، ساخت مدل ترکیبی وزن‌دار، و ارائه توضیح‌پذیری با SHAP است. پایداری روش نیز با میانگین F1 برابر ۰/۰۱۷ ± ۰/۹۷۵در اعتبارسنجی متقاطع ده‌تایی تأیید شده است. شکل‌های ۹ و ۱۰ این عملکرد مناسب در صحت و یادآوری را به‌صورت بصری نشان می‌دهند و تأیید می‌کنند که روش پیشنهادی با رفع محدودیت‌های اساسی مطالعات پیشین و برتری به نسبت آنها، رویکردی جامع‌تر و کاربردی‌تر برای پیش‌بینی نقص نرم‌افزار ارائه می‌دهد.

۴-۵ کارهای آینده

برای بهبود و گسترش روش پیشنهادی، چندین مسیر پژوهشی پیشنهاد می‌شود. نخست، بهینه‌سازی زمان اجرا از طریق استفاده از الگوریتم‌های موازی، کاهش تعداد تکرارها در Optuna، یا بهره‌گیری از تکنیک‌های بهینه‌سازی کارآمدتر می‌تواند کاربرد این روش را در محیط‌های عملیاتی با منابع محدود یا نیاز به پاسخ‌دهی سریع گسترش دهد.

دوم، آزمایش روش پیشنهادی روی مجموعه داده‌های متنوع‌تر از پروژه‌های مختلف (مانند CM1، JM1، PC1، و دیگر مجموعه داده‌های مخزن PROMISE)، زبان‌های برنامه‌نویسی گوناگون (مانند Python، Java، یا JavaScript)، و حوزه‌های کاربردی متفاوت (مانند سیستم‌های تعبیه‌شده، برنامه‌های وب، یا نرم‌افزارهای موبایل) می‌تواند تعمیم‌پذیری و قابلیت انطباق آن را تأیید کند و محدودیت‌های احتمالی در کاربردهای خاص را مشخص سازد.

سوم، ادغام تکنیک‌های یادگیری عمیق نظیر شبکه‌های عصبی عمیق، شبکه‌های هم‌آمیختی، شبکه‌های بازگشتی[47]، یا مدل‌های مبتنی بر مبدّل[48] (مانند CodeBERT یا GraphCodeBERT) می‌تواند توانایی روش در ثبت الگوهای پیچیده‌تر، وابستگی‌های بلندمدت در کد، یا روابط ساختاری بین اجزای نرم‌افزار را افزایش دهد. این تکنیک‌ها به‌ویژه در

47-RNN
48-Transformer

پروژه‌های بزرگ با کد پیچیده می‌توانند مفید باشند.

چهارم، توسعه ابزارهای بصری‌سازی تعاملی و داشبوردهای کاربرپسند برای تحلیل SHAP و نمایش نتایج پیش‌بینی می‌تواند فهم نتایج را برای توسعه‌دهندگان، آزمایش‌کنندگان، و مدیران پروژه آسان‌تر کرده و تصمیم‌گیری‌های مبتنی بر داده را تسهیل کند. این ابزارها می‌توانند شامل نمودارهای تعاملی، گزارش‌های خودکار، و سیستم‌های هشدار زودهنگام برای ماژول‌های پرخطر باشند.

پنجم، بررسی اثر انتخاب ویژگی‌های تطبیقی مختلف، ترکیب‌های متفاوت از عملیات ریاضی (مانند تبدیل‌های توانی، ریشه‌گیری، یا نرمال‌سازی‌های پیشرفته‌تر)، و روش‌های انتخاب ویژگی (مانند انتخاب مبتنی بر اطلاعات متقابل یا الگوریتم‌های ژنتیک) بر عملکرد روش و شناسایی ترکیب‌های بهینه برای پروژه‌های خاص می‌تواند انعطاف‌پذیری و کارایی رویکرد را بهبود بخشد.

ششم، بررسی اثر متعادل‌سازی با روش‌های دیگر (مانند ADASYN[49]، SMOTE، یا ترکیب‌های مختلف Over–Sampling و Under-Sampling) و مقایسه آن‌ها با NearMiss می‌تواند به شناسایی بهترین راهبرد برای انواع مختلف داده‌های نامتعادل کمک کند.

در نهایت، توسعه یک سیستم یکپارچه و خودکار برای پیش‌بینی نقص نرم‌افزار که بتواند به‌صورت پیوسته در طول چرخه توسعه نرم‌افزار اجرا شود، به توسعه‌دهندگان بازخورد لحظه‌ای ارائه دهد، و با ابزارهای توسعه (مانند IDEها، سیستم‌های کنترل نسخه، یا سیستم‌های CI/CD) یکپارچه شود، می‌تواند ارزش عملی این پژوهش را به‌طور قابل‌توجهی افزایش دهد.

این پیشنهادها، با تمرکز بر بهبود کارایی، گسترش کاربردها، و افزایش قابلیت استفاده عملی، می‌توانند روش پیشنهادی را به رویکردی کاربردی‌تر و گسترده‌تر در حوزه پیش‌بینی نقص نرم‌افزار تبدیل کنند و به توسعه‌دهندگان در شناسایی زودهنگام و مدیریت بهتر نقص‌های نرم‌افزاری کمک کنند.

49-Adaptive Synthetic Algorithm

۵- نتیجه‌گیری

در حوزه مهندسی نرم‌افزار، پیش‌بینی نقص نرم‌افزاری به عنوان ابزاری کلیدی برای کاهش ریسک‌های عملیاتی و بهینه‌سازی فرایندهای توسعه عمل می‌کند، جایی که شناسایی زودهنگام نقص‌ها می‌تواند از هزینه‌های هنگفت نگهداری و اختلالات سیستم جلوگیری نماید. رویکرد پیشنهادی یادگیری ماشین ترکیبی خودکار با تکامل ویژگی تطبیقی، با ادغام مراحل پیش‌پردازش داده‌ها (شامل حذف ویژگی‌های بی‌واریانس، پر کردن با میانه، و نرمال‌سازی)، تولید ویژگی‌های تطبیقی (مانند نسبت‌های متریک و ترکیب‌های تعاملی)، متعادل‌سازی کلاس‌ها با NearMiss (نسخه ۱)، بهینه‌سازی چندهدفه با Optuna، و ساخت مدل ترکیبی وزن‌دار، رویکردی یکپارچه ارائه می‌دهد که چالش‌های سنتی مانند عدم تعادل کلاس‌ها و کمبود توضیح‌پذیری را برطرف می‌سازد.

نتایج ارزیابی روی مجموعه داده KC1 از مخزن PROMISE ناسا، با دستیابی به دقّت ۹۶/۹ درصد، معیار F1 برابر ۹۶/۸درصد، یادآوری ۹۳/۸ درصد و AUC برابر ۹۷/۲ درصد، برتری قابل‌توجهی نسبت به مدل‌های پایه (مانند گرادیان بوستینگ با یادآوری تنها ۲۳/۱ درصد) و مطالعات پیشین نشان می‌دهد. تمرکز بر یادآوری بالا، که نسبت نقص‌های صحیح شناسایی‌شده به کل نقص‌ها را اندازه‌گیری می‌کند، اهمیت عملی رویکرد را برجسته می‌سازد، زیرا عدم تشخیص نقص‌ها می‌تواند به خرابی‌های پرهزینه منجر شود. علاوه بر این، تحلیل توضیح‌پذیری با SHAP TreeExplainer، با محاسبه مقادیر Shapley و شناسایی ویژگی‌های کلیدی مانند تعداد خطوط کد (loc با اهمیت ۵/۰۴) و معیار t (با اهمیت ۳/۷۷)، امکان تفسیر تصمیم‌گیری‌های مدل را فراهم می‌کند و مدل‌های جعبه‌سیاه سنتی را به ابزاری شفاف و کاربردی تبدیل می‌نماید. پایداری رویکرد نیز با میانگین F1 برابر ۰/۹۷۵ ± ۰/۰۱۷ در اعتبارسنجی متقاطع ده‌تایی تأیید شد، که نشان‌دهنده قابلیت اعتماد آن در سناریوهای واقعی است.

با وجـــود ایـــن دســـتاوردها، رویکرد پیشـــنهادی با محدودیت‌هـــایی ماننـــد پیچیـــدگی محاســـباتی (ناشی از بهینه‌سازی و تحلیل SHAP) و تمرکز بر یک مجموعه داده خاص مواجه اســـت، که ممکن است در محیط‌های با منابع محدود یا داده‌های ناهمگن چالش‌برانگیز باشد. برای غلبه بر این‌ها، پیشـــنهاد می‌شود در کارهای آینده، بهینه‌سازی زمـــان اجرا از طریق الگوریتم‌های مـــوازی، ارزیابی روی مجموعه‌های متنوع‌تر مانند CM1 و JM1، و ادغام تکنیک‌های یادگیری عمیق مانند شبکه‌های هم‌آمیختی پیگیری شود.

در نهایـــت، این پژوهـــش با ارائه رویکـــردی جامع که دقت، پایداری، و توضیح‌پذیـــری را هم‌زمان ارتقا می‌دهد، به توســـعه‌دهندگان کمک می‌کند تا ریسک‌های نرم‌افزاری را مدیریت کنند و کیفیت کلی سیســـتم‌ها را بهبود بخشند، و پتانســـیـــل بالایی برای کاربـــرد در پروژه‌های صنعتی و پژوهشی آینده دارد.

منابع

1. Wahono, R. S. (2015). “A systematic literature review of software defect prediction,” Journal of software engineering, vol. 1, no. 1, pp. 1–16.
2. Matloob, F., et al., (2021). “Software defect prediction using ensemble learning: A systematic literature review,” IEEE Access, vol. 9, pp. 98754–98771, doi: 10.1109/ACCESS.2021.3095559.
3. Malhotra, R. (2015). “A systematic review of machine learning techniques for software fault prediction,” Appl Soft Comput, vol. 27, pp. 504–518, Feb. 2015, doi: 10.1016/J.ASOC.2014.11.023.
4. Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). “Benchmarking classification models for software defect prediction: A proposed framework and novel findings,” IEEE Transactions on Software Engineering, vol. 34, no. 4, pp. 485–496, doi: 10.1109/TSE.2008.35.
5. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M. & Hutter, F. (2015). “Efficient and robust automated machine learning,” Adv Neural Inf Process Syst, vol. 28.
6. Basgalupp, M. P., R. C. Barros, T. S. Da Silva, F. F. Silveira, P. B. C. Miranda, and F. Neri, “An Experimental Analysis on Automated Machine Learning for Software Defect Prediction,” 2024 IEEE Congress on Evolutionary Computation, CEC 2024 - Proceedings, 2024, doi: 10.1109/CEC60901.2024.10611946.
7. Qiu, S., He, B. E. J., & Liu, L. (2024). “Survey of software defect prediction features,” Neural Computing and Applications 2024 37:4, vol. 37, no. 4, pp. 2113–2144, Dec. 2024, doi: 10.1007/S00521-024-10937-1.
8. Thota, M. K., Shajin, F. H., & Rajesh, P. (2020). “Survey on software defect prediction techniques,” International Journal of Applied Science and Engineering, vol. 17, no. 4, pp. 331–344.
9. Arar, Ö. F., & Ayan, K. (2015). “Software defect prediction using cost-sensitive neural network,” Appl Soft Comput, vol. 33, pp. 263–277, Aug. 2015, doi: 10.1016/J.ASOC.2015.04.045.
10. Bahaweres, R. B., Agustian, F., Hermadi, I., Suroso, A. I., & Arkeman, Y. (2020). “Software defect prediction using neural network based smote,” International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), vol. 2020-October, pp. 71–76, Oct. 2020, doi: 10.23919/EECSI50503.2020.9251874.
11. Alkhasawneh, M. S. (2022). “Software Defect Prediction through Neural Network and Feature Selections,” Applied Computational Intelligence and Soft Computing, vol. 2022, no. 1, p. 2581832, Jan. 2022, doi: 10.1155/2022/2581832.
12. Mehmood et al., (2023). “A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning,” IEEE Access, vol. 11, pp. 63579–63597, doi: 10.1109/ACCESS.2023.3287326.
13. Mcmurray, S. & Sodhro, A. H. (2023). “A Study on ML-Based Software Defect Detection for Security Traceability in Smart Healthcare Applications,” Sensors 2023, Vol. 23, Page 3470, vol. 23, no. 7, p. 3470, Mar. 2023, doi: 10.3390/S23073470.
14. Eka Yulia Retnani, W., Furqon, A., Setiawan, J., & Y Retnani, W. E. (2024). “Improving Software Defect Prediction Using a Combination of Ant Colony Optimization-based Feature Selection and Ensemble Technique,” Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, vol. 9, no. 4, pp. 389–396, Nov. 2024, doi: 10.22219/KINETIK.V9I4.2038.
15. Shrimankar, R., Kuanr, M., Piri, J., & Panda, N. (2022). “Software Defect Prediction: A Comparative Analysis of Machine Learning Techniques,” Proceedings - 2022 International Conference on Machine Learning, Computer Systems and Security, MLCSS 2022, pp. 38–47, 2022, doi: 10.1109/MLCSS57186.2022.00016.
16. Babatunde, N., Ogundokun, R. O., Adeoye, L. B., & Misra, S. (2023). “Software Defect Prediction Using Dagging Meta-Learner-Based Classifiers,” Mathematics 2023, Vol. 11, Page 2714, vol. 11, no. 12, p. 2714, Jun. 2023, doi: 10.3390/MATH11122714.
17. Bluemke, & Borsukiewicz, P. (2023) “Experiments on Software Error Prediction Using Decision Tree and Random Forest Algorithms,” Proceedings of the 18th Conference on Computer Science and Intelligence Systems, FedCSIS 2023, pp. 865–869, 2023, doi:

10.15439/2023F363.
18. Siddika, M., & Begum, F. (2025). Al Farid, J. Uddin, and H. A. Karim, "Enhancing Software Defect Prediction Using Ensemble Techniques and Diverse Machine Learning Paradigms," Eng 2025, Vol. 6, Page 161, vol. 6, no. 7, p. 161, Jul. 2025, doi: 10.3390/ENG6070161.
19. Shankar, S. P., & Chaudhari, S. S. (2024). "Analysis of Bio Inspired Based Hybrid Learning Model for Software Defect Prediction," SN Comput Sci, vol. 5, no. 7, pp. 1–22, Oct. 2024, doi: 10.1007/S42979-024-03171-Y/METRICS.
20. Das, M., Prasad, N., & Mohan, B. R. (2025). "Improving Machine Learning Models with Hybrid Metaheuristic Algorithm for Software Defect Prediction," Communications in Computer and Information Science, vol. 2461 CCIS, pp. 44–61, 2025, doi: 10.1007/978-3-031-96473-2_4.
21. Abdu et al., (2025). "Cross-project software defect prediction based on the reduction and hybridization of software metrics," Alexandria Engineering Journal, vol. 112, pp. 161–176, Jan. 2025, doi: 10.1016/J.AEJ.2024.10.034.
22. Sánchez-García et al., M. J. , (2023). "Software Defect Prediction with Bayesian Approaches," Mathematics 2023, Vol. 11, Page 2524, vol. 11, no. 11, p. 2524, May 2023, doi: 10.3390/MATH11112524.
23. Kumar, D., Yadav, K., & Prasad, M. (2025). "Predicting fault-prone software modules using bayesian belief network: an empirical study," International Journal of System Assurance Engineering and Management 2025 16:6, vol. 16, no. 6, pp. 2204–2218, May 2025, doi: 10.1007/S13198-025-02809-1.
24. Benala, T. R., & Tantati, K. (2022). "Efficiency of oversampling methods for enhancing software defect prediction by using imbalanced data," Innovations in Systems and Software Engineering 2022 19:3, vol. 19, no. 3, pp. 247–263, Jun. 2022, doi: 10.1007/S11334-022-00457-3.
25. Alam, M., & Mustaqeem, M. (2024). "Reinforcing defect prediction: a reinforcement learning approach to mitigate class imbalance in software defect prediction," Iran Journal of Computer Science 2024 8:1, vol. 8, no. 1, pp. 151–162, Nov. 2024, doi: 10.1007/S42044-024-00214-8.
26. Kumar, H., Saxena, V., Kumar, H., & Saxena, V. (2024). "Software Defect Prediction Using Hybrid Machine Learning Techniques: A Comparative Study," Journal of Software Engineering and Applications, vol. 17, no. 4, pp. 155–171, Apr. 2024, doi: 10.4236/JSEA.2024.174009.
27. Iqbal et al., (2019). "Performance analysis of machine learning techniques on software defect prediction using NASA datasets," International Journal of Advanced Computer Science and Applications, Vol. 10, No. 5.
28. Sakib, F. F., Islam Juel, M. S., Tafannum, Z., Joy, J. S., & Karim, R. (2025). "A Comprehensive Evaluation of Machine Learning Techniques for Predicting Software Bugs," International Conference on Robotics, Electrical and Signal Processing Techniques, pp. 228–233, 2025, doi: 10.1109/ICREST63960.2025.10914387.
29. Hall, T., Beecham, S., Bowes, D., Gray, D. & Counsell, S. (2012). "A systematic literature review on fault prediction performance in software engineering," IEEE Transactions on Software Engineering, vol. 38, no. 6, pp. 1276–1304, 2012, doi: 10.1109/TSE.2011.103.
30. Guyon & Elisseeff, A. (2003). "An introduction to variable and feature selection," Journal of machine learning research, vol. 3, no. Mar, pp. 1157–1182.
31. Mining, W. I. D. (2006). "Data mining: Concepts and techniques," Morgan Kaufinann, vol. 10, no. 559–569, p. 4, 2006.
32. Jolliffe, (2011). "Principal component analysis," in International encyclopedia of statistical science, Springer, 2011, pp. 1094–1096.
33. Han, H., Wang, W.-Y. & Mao, B.-H. (2005). "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," in International conference on intelligent computing, Springer, 2005, pp. 878–887.
34. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). "Optuna: A next-generation hyperparameter optimization framework," in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2623–2631.
35. Chen, T. & Guestrin, C. (2016). "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.
36. Kohavi, R. (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection," in Ijcai, Montreal, Canada, 1995, pp. 1137–1145.
37. Shepperd, M. & Song, Q., Sun, Z. & Mair, C. (2013). "Data quality: Some comments on the NASA software defect datasets," IEEE Transactions on Software Engineering, vol. 39, no. 9, pp. 1208–1215, 2013, doi: 10.1109/TSE.2013.11.
38. "GitHub - ApoorvaKrisna/NASA-promise-dataset-repository." Accessed: Aug. 29, 2025. [Online]. Available: https://github.com/ApoorvaKrisna/NASA-promise-dataset-repository