

ارائه یک روش نوین جهت پیش‌بینی نقص نرم‌افزار با استفاده از ترکیب شبکه عصبی و الگوریتم ملخ

سمیه شعبانی زاده رابری^۱، وحید خطیبی بردسیری^{۲*} و عمید خطیبی بردسیری^۳

| اطلاعات مقاله | چکیده |
|---|---|
| دریافت مقاله: ۹۷/۰۵/۱۲ | |
| پذیرش مقاله: ۹۷/۱۱/۲۴ | |
| واژگان کلیدی: | |
| قابلیت اطمینان، پیش‌بینی نقص نرم‌افزار، شبکه‌های عصبی پرسپترون، الگوریتم ملخ. | سیکل توسعه نرم‌افزار شامل آنالیز، طراحی، پیاده‌سازی و تست و یکسری فازهای دیگر است. مرحله تست نرم‌افزار یکی از مراحل هزینه‌بر توسعه نرم‌افزار است. باید به‌طور مؤثری انجام شود تا نرم‌افزار بدون خطا دست کاربران برسد. یکی از فعالیت‌های مؤثر برای توسعه نرم‌افزار و افزایش قابلیت اطمینان آن، پیش‌بینی نقص نرم‌افزار قبل از رسیدن به مرحله تست است که کمک حائز اهمیت برای صرفه‌جویی زمانی در فرآیند تولید، نگهداری و هزینه آن می‌کند. یکی از مدل‌های کارا برای پیش‌بینی نقص نرم‌افزار، استفاده از شبکه‌های عصبی پرسپترون چندلایه با الگوریتم آموزشی پس‌انتشار خطا است. یکی از نقاط ضعف الگوریتم آموزشی پس‌انتشار خطا احتمال به دام افتادن شبکه عصبی در نقاط مینیمم محلی است. با توجه به قابلیت الگوریتم‌های فراابتکاری در خروج از دام مینیمم‌های محلی و یافتن مینیمم سراسری، در این مقاله جهت برطرف کردن ضعف الگوریتم آموزشی شبکه عصبی و بهبود دقت آن در زمینه پیش‌بینی نقص نرم‌افزار، از ترکیب الگوریتم فراابتکاری ملخ با الگوریتم آموزشی پس‌انتشار خطا استفاده شده است. جهت ارزیابی نتایج حاصل از مدل پیشنهادی، نه پایگاه داده واقعی بکار گرفته شده و روش ارزیابی متقاطع مبنای ارائه نتایج بوده است. عملکرد مدل پیشنهادی با شش مدل پیش‌بینی نقص نرم‌افزار مقایسه شده است. نتایج این مقایسه نشان می‌دهد که مدل پیشنهادی قادر است در تعداد زیادی از مجموعه داده، صحت و دقت بالاتری نسبت به سایر مدل‌ها ارائه دهد. |

۱- مقدمه

آزمون نرم‌افزار یکی از اساسی‌ترین فازهای توسعه آن به شمار می‌رود. پروژه NOTO-Rious مثالی از اهمیت آزمایش و پیش‌بینی نقص نرم‌افزار است. NOTO-Rious نام یکی از فضاپیمای ناسا است که در آن به دلیل یک اشکال کوچک تبدیل داده؛ ۱۲۵ میلیون دلار از دست رفت. این مثال نشان می‌دهد که رفع اشکال نرم‌افزار در حین توسعه آن، خیلی مقرون به‌صرفه‌تر است تا بعد از استقرار نرم‌افزار [۱]. در پروژه‌های نرم‌افزاری کشف و اصلاح نقص در بازه‌های زمانی

موجود بسیار وقت‌گیر و در توسعه نرم‌افزار بسیار هزینه‌بر است، به همین علت تأثیر تست و بازیابی کدها کاهش می‌یابد. از این‌رو شناسایی ماژول‌های مستعد خطا به‌صورت خودکار، به تمرکز روی تست و بازیابی کدها، کمک می‌کند. بنابراین جمع‌آوری دقیق داده‌های مستعد خطا، نیاز به ابزاری جهت پیش‌بینی ماژول‌های مستعد خطا دارد [۲].

تشخیص مؤثر نقص‌های نرم‌افزار در واقع یک فعالیت مهم از فرآیند توسعه نرم‌افزار می‌باشد، که به بهبود و دقت

* پست الکترونیک نویسنده مسئول: khatibi78@yahoo.com

۱. گروه علمی مهندسی کامپیوتر، موسسه آموزش عالی غیرانتفاعی غیردولتی جاوید، جیرفت

۲. استادیار گروه مهندسی کامپیوتر، واحد بردسیر، دانشگاه آزاد اسلامی، بردسیر، ایران

۳. استادیار گروه مهندسی کامپیوتر، واحد بردسیر، دانشگاه آزاد اسلامی، بردسیر، ایران

بودن ماژول نرم‌افزاری از طریق روش‌های مبتنی بر استخراج قوانین انجمنی تشخیص داده می‌شود [۷]. اصلی-ترین روش پیش‌بینی نقص نرم‌افزار، روش سوم است. در روش طبقه‌بندی (روش سوم) از روش‌های یادگیری ماشین استفاده می‌شود. در روش طبقه‌بندی با توجه به سابقه ماژول ناقص، مدلی استخراج می‌شود که بر اساس آن بتوان با دقتی مناسب پیش‌بینی نقص در ماژول جدید را انجام داد. پژوهش‌های قبلی بر اساس یادگیری از متریک‌های به‌دست‌آمده از ماژول‌های نرم‌افزاری نشان می‌دهد که ارتباطی قوی بین این متریک‌ها و پیش‌بینی نقص وجود دارد [۸]. ادامه این مقاله در هشت قسمت سازمان‌دهی شده است: بخش دوم به مرور کارهای مرتبط می‌پردازد. بخش سه و چهار و پنج به ترتیب الگوریتم ملخ و دیتاست، شبکه عصبی را توضیح می‌دهند. مدل پیشنهادی در بخش ششم آورده شده و بخش هفت و هشت نیز به ترتیب نحوه ارزیابی و نتایج به دست آمده را نشان می‌دهند. در نهایت بخش نهم به نتیجه‌گیری و پیشنهاد کارهای آینده اختصاص دارد.

۲- کارهای مرتبط

روش‌های مختلف مدل‌سازی پیش‌بینی خطا در نرم‌افزار به دودسته تقسیم شده‌اند:

- روش‌های آماری^۱ (الگوریتمی)
- روش‌های یادگیری ماشین^۲ (غیر الگوریتمی)

روش‌های آماری، از توزیع‌های آماری، برای پیش‌بینی خطای نرم‌افزار استفاده می‌کنند. احتمال بروز خطا در یک ماژول نرم‌افزاری را محاسبه می‌نمایند [۹]. در روش‌های یادگیری ماشین، از الگوریتم‌های یادگیری ماشین، استفاده می‌شود. برای پیش‌بینی خطای نرم‌افزار، الگوریتم یادگیری ماشین را روی دادگان ارزیابی^۳ مخصوص خطا در نرم‌افزار، پیاده‌سازی و اجرا کرده و با توجه به معیارهای ارزیابی^۴ دقت، نتایج را می‌سنجند. آزمون نرم‌افزار، یک فعالیت بسیار پیچیده در طول تولید نرم‌افزار است. مدل‌های پیش‌بینی به متخصصان در این مورد با استفاده از روش‌های یادگیری ماشین، کمک کرده‌اند [۱۰]. طبقه‌بندی شکلی از تحلیل داده‌ها محسوب می‌شود که می‌توان به کمک آن مدلی

پیش‌بینی نقص می‌تواند کمک بسزایی کند. به همین دلیل پیش‌بینی نقص نرم‌افزار یک موضوع تحقیقاتی مهم در مهندسی نرم‌افزار بیان شده است. با یافتن مدل‌های مطلوب برای پیش‌بینی نقص نرم‌افزار اغلب تعدادی از نقص‌های توزیع‌شده را مورد تجزیه و تحلیل قرار می‌دهند [۳]. بنا به یک گزارش منتشرشده در سال ۲۰۱۳، ۳٫۷ تریلیون دلار در سراسر جهان صرف هزینه‌های نرم‌افزار می‌شود. بررسی نشان داد که ۲۳ درصد از این هزینه در تضمین کیفیت و آزمودن صرف شده است [۱۱]. گاهی رخ دادن خطا در نرم‌افزار می‌تواند باعث ضررهای مالی و حتی در برخی کاربردها باعث ضررهای جانی گردد. به همین دلیل تضمین قابلیت اطمینان به عنوان یک نیازمندی غیرکارکردی (کیفی) از اهمیت بسزایی برخوردار است [۴]. اگر خطای موجود در فرآیند را زود هنگام بیابید، تصحیح آن، هزینه کمتری در بر خواهد داشت. پیدا کردن و رفع خطا، مستلزم هزینه بالای است. داده‌های منتشرشده از وزارت دفاع آمریکا نشان می‌دهد که در سال ۲۰۰۶ آمریکا حدود ۷۸۰ بلیون دلار را صرف خطاهای نرم‌افزاری کرده است. پیش‌بینی خطای نرم‌افزار، مدیران نرم‌افزار را قادر می‌سازد که تلاش برای بهبود کیفیت نرم‌افزار را در قسمت‌های مورد نیاز متمرکز کنند. به همین منظور روش‌های مختلف مدل‌سازی پیش‌بینی خطای نرم‌افزار توسعه داده شده‌اند و به‌طور واقعی برای پیش‌بینی کیفیت نرم‌افزار مورد استفاده قرار می‌گیرند [۵]. پیش‌بینی نقص نرم‌افزار روی روش‌های الف- تخمین تعداد نقص‌های باقیمانده در سیستم نرم‌افزاری؛ ب- کشف روابط موجود بین نقص‌ها؛ ج- طبقه‌بندی مؤلفه‌های نرم‌افزار در دو کلاس مستعد نقص و غیرمستعد نقص، تمرکز دارد [۶]. در روش اول در مراحل طراحی و نیازمندی، اسنادی توسط مهندسان نرم‌افزاری که ویژگی‌های طراحی و نیازمندی را شرح می‌دهند آماده می‌شود. این اسناد توسط چندین (۳ تا ۵) کارشناس بررسی می‌شود. هر کدام از کارشناس‌ها قبل از تأیید نهایی اسناد به مسائلی که باید حل شوند (با توجه به اعتقادشان) توجه می‌کنند. فهرستی از تمام بررسی‌های انجام‌شده تهیه می‌شود. اگر میزان همپوشانی مسائلی که در بررسی‌های انجام‌شده کشف نشده‌اند، کوچک باشد اجازه رفتن به مرحله بعدی داده می‌شود. در روش دوم ناقص یا غیرناقص

³ Datasets

⁴ Evaluation Measures

¹ Statistical Method

² Machine Learning Method

است تا فرآیند کلاس‌بندی دقیق‌تر انجام شود، داده می‌شود. سپس کیفیت نرم‌افزار به کمک قابلیت نگهداری و قابلیت اطمینان اندازه‌گیری می‌شود. از تجزیه و تحلیل مقایسه روشن است که روش پیشنهادی کیفیت بهتری در مقایسه با سایر روش‌های موجود به دست آورده است [۲۳]. پال^۷ و همکاران، دو مدل پیشنهادی، پیش‌بینی خطای نرم‌افزار به کمک شبکه عصبی مصنوعی ارائه کردند. شبکه عصبی دو روش ترکیبی پیاده‌سازی شده است: الف- شبکه عصبی با استفاده از الگوریتم ژنتیک ب- شبکه عصبی با استفاده از الگوریتم جفت‌گیری پرندگان. مقایسه‌ها به‌طور واضح نشان می‌دهند که صحت و دقت مدل پیشنهادی، ترکیب شبکه عصبی با الگوریتم جفت‌گیری پرندگان بهتر از مدل پیشنهادی، ترکیب شبکه عصبی با الگوریتم ژنتیک است [۲۴]. ماشین بردار پشتیبان یک روش طبقه‌بندی پیشرفته‌ای هست که برای طبقه‌بندی خطا مناسب است. رانگ^۸ و همکاران، مدل پیش‌بینی خطای نرم‌افزار با ترکیب ماشین بردار پشتیبان و الگوریتم خفاش ارائه کردند. این مدل پیشنهادی باعث بهبود توانایی محاسبات غیرخطی SVM و قابلیت بهینه‌سازی پارامترهای الگوریتم خفاش می‌شود. نتایج نشان می‌دهد، مدل ترکیبی SVM با تغییر رنج الگوریتم خفاش برای پیش‌بینی خطای نرم‌افزار، میزان صحت بیشتری در مقایسه با مدل ماشین بردار پشتیبان و دیگر مدل‌های سنتی دارد [۲۵]. هدف اصلی مقاله راجیکا^۹ و همکار، عبارت است از: الف) انتخاب ویژگی برای پیش‌بینی نقص با استفاده از الگوریتم تکاملی پیشنهادی جهت کاهش ویژگی (ب) مقایسه تکنیک‌های یادگیری ماشین (ج) استفاده از دقت و فراخوانی به عنوان معیار کارایی برای پیش‌بینی نقص (د) اعتبار دهی ۱۰ بار بر روی هر مدل انجام شده است. در این استراتژی، کلاس خطا دار را با استفاده از ۵ تکنیک یادگیری ماشین و ۲ تکنیک الگوریتم تکاملی برای انتخاب ویژگی پیش‌بینی شده است. الگوریتم‌های تکاملی را برای انتخاب ویژگی مناسب برای هر یک از تکنیک‌های طبقه‌بندی اعمال شده در پنج بسته آندروید منبع باز استفاده شده است. نتایج نشان می‌دهد مدل‌های پیش‌بینی که از الگوریتم‌های تکاملی برای انتخاب ویژگی

جهت توصیف داده‌ها استخراج کرد. با کمک این مدل می‌توان نمونه‌ها را به یکی از چندین طبقه تعریف‌شده انتساب داد. فرآیند ساخت مدل فرایندی دومارحله‌ای است. در مرحله اول با کمک دادگان آموزش که برچسب تمام نمونه‌های آن مشخص است مدل ساخته می‌شود (مرحله یادگیری). در مرحله دوم با کمک دادگان آزمون مدل به‌دست‌آمده اعتبار سنجی می‌شود. ارزشیابی مدل با توجه به اینکه کلاس چه تعداد از نمونه‌های آزمون را درست تخمین زد است، محاسبه می‌شود. در حوزه پیش‌بینی نقص نرم‌افزار طبقه‌بندی‌های مختلفی از جمله رگرسیون منطقی [۱۱-۱۳]، جنگل تصادفی [۱۴، ۱۵]، نایویز [۱۴، ۱۶-۱۸]، شبکه عصبی^۱ MLP [۱۸، ۱۹]، SVM^۲ و توسعه‌های آن [۲۰] و KNN^۳ [۱۲، ۲۰] بکار گرفته می‌شود. شبکه عصبی یکی از پرکاربردترین روش‌ها جهت پیش‌بینی خطا در نرم‌افزار است. زانگ^۴ و همکاران، روش پیش‌بینی خطای نرم‌افزاری جدید بر اساس شبکه عصبی خاکستری معرفی کردند، در این مطالعه بیشتر تمرکز روی مسئله کاهش خطا و تکنیک جبران خطا بود [۲۱] آرار^۵ و همکاران در استراتژی خود ترکیبی از شبکه عصبی مصنوعی و الگوریتم کلونی زنبور عسل استفاده کردند. پیاده‌سازی الگوریتم پیشنهادی در محیط متلب و بر روی پروژه‌های cm1, kc1, kc2, pc1, jm1 از مخزن ناسا انجام شده است و نتایج پیاده‌سازی نشان‌دهنده برتری الگوریتم پیشنهادی از لحاظ هزینه و دقت و صحت نسبت به الگوریتم پس‌انتشار خطا است [۱]. الگوریتم‌های هوشمند مانند الگوریتم ژنتیک توانایی بالایی در پیش‌بینی دارد، می‌توان از آن‌ها در پیش‌بینی وضعیت آتی نرم‌افزار یا پیش‌بینی خطای نرم‌افزاری استفاده نمود. فاضل روشی برای پیش‌بینی خطای نرم‌افزار با استفاده از الگوریتم ژنتیک ارائه داده است. هدف از ارائه این روش پیش‌بینی خطای نرم‌افزار با سرعت و دقت بالاتر است و همچنین ارائه ساختاری که به‌سادگی قابل پیاده‌سازی و تعمیم باشد. [۲۲]. شران^۶ و همکاران، یک مدل برای پیش‌بینی بهتر کیفیت نرم‌افزار با استفاده از طبقه‌بند شبکه عصبی ارائه داده‌اند. موارد آزمون به طبقه‌بندی که با الگوریتم بهینه‌ساز فاخته ترکیب شده

⁶ Sheoran⁷ Pal⁸ Rong⁹ Ruchika¹ Multi-layer Perceptron² Support Vector Machine³ Nearest Neighbor⁴ Zhang⁵ Arar

تابع S که نیروی اجتماعی را تعریف می‌کند طبق رابطه (۴) محاسبه می‌شود: که در آن f نشان‌دهنده شدت جاذبه و l نشان‌دهنده طول نشان‌دهنده طول مقیاس جاذبه است.

$$s(r) = fe^{-\frac{r}{l}} - e^{-r} \quad (4)$$

که در آن f نشان‌دهنده شدت جاذبه و l نشان‌دهنده طول مقیاس جاذبه می‌باشد. اگرچه تابع S قادر است تا فضای بین دو ملخ را به نواحی دافعه و جاذبه و آسایش تقسیم کند، این تابع مقادیر نزدیک به صفر بافاصله بیشتر از ۱۰ همانند آنچه در شکل (۱) نشان داده شده است را برمی‌گرداند. بنابراین، این تابع قادر نیست تا نیروی شدیدی را بین دو ملخ که فاصله بین آن‌ها زیاد است را اعمال کند. برای حل این مشکل، ما فاصله بین دو ملخ را به بازه $[۱, ۴]$ نگاشت می‌کنیم.

مؤلفه G در رابطه (۵) به صورت زیر محاسبه می‌شود:

$$G_i = -g\hat{e}_g \quad (5)$$

که g ثابت گرانشی زمین و \hat{e}_g بردار واحد به سمت مرکز زمین می‌باشد.

مؤلفه A در رابطه (۶) به صورت زیر محاسبه می‌شود:

$$A_i = u\hat{e}_w \quad (6)$$

U ثابت رانش و \hat{e}_w بردار واحد در جهت باد می‌باشد. ملخ‌های پوره بال ندارند، بنابراین حرکت آن‌ها به شدت به جهت وزش باد بستگی دارد.

جایگزینی S و G و A در رابطه (۷) به صورت زیر می‌باشد:

$$X_i = \sum_{j \neq i}^N s(|x_j - x_i|) \frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\hat{e}_w \quad (7)$$

در رابطه بالا $s(r) = fe^{-\frac{r}{l}} - e^{-r}$ و N تعداد ملخ‌هاست. رابطه (۷) مانع کاوش و بهره‌برداری در فضای جستجوی اطراف یک راه‌حل می‌شود. نمی‌توان به‌طور مستقیم از این رابطه برای حل مسائل بهینه‌سازی مورد استفاده قرار داد. به همین دلیل یک نسخه اصلاح‌شده از این رابطه برای حل مسائل بهینه‌سازی به صورت رابطه (۸) ارائه شده است:

$$X_i^d = c \left(\sum_{j \neq i}^N c \frac{ub_d - lb_d}{2} s(|x_j^d - x_i^d|) \frac{x_j^d - x_i^d}{d_{ij}} \right) + \hat{T}_d \quad (8)$$

در معادله بالا ub_d حد بالا و lb_d حد پایین در بعد d ام می‌باشد $s(r) = fe^{-\frac{r}{l}} - e^{-r}$ و \hat{T}_d مقدار بعد d ام در هدف (بهترین جوابی که تاکنون دیده شده است)

استفاده شده است، قدرت پیش‌بینی کننده بهتری نسبت به مدل‌هایی که از تکنیک انتخاب ویژگی استفاده نکرده دارد [۲۶].

در مطالعات انجام‌شده، کاربردی از الگوریتم ملخ دیده نمی‌شود. با توجه به اینکه الگوریتم ملخ یکی از محبوب‌ترین الگوریتم‌های بهینه‌سازی مبتنی بر چند راه‌حل است که در سال ۲۰۱۷ معرفی شده است. دارای فرآیند اکتشاف قوی‌تر، مکانیسم جستجوی قوی در فضای مسئله، بهینه‌سازی با سرعت بیشتر انجام می‌دهد. در ترکیب با شبکه عصبی کارایی بهینه‌تری دارد [۲۷]. در مدل پیشنهادی ترکیبی از شبکه عصبی و الگوریتم ملخ استفاده شده است.

۳- الگوریتم ملخ

الگوریتم ملخ نیز مانند سایر الگوریتم‌های الهام گرفته‌شده از طبیعت، به صورت منطقی فرآیند جستجو را به دو بخش تقسیم می‌کند: اکتشاف و بهره‌برداری. در اکتشاف، عامل‌های جستجو تشویق به حرکت‌های تصادفی می‌شوند. در مرحله بهره‌برداری آن‌ها تمایل به حرکت‌های محلی و اطراف مکان خود دارند. این دو عمل و همچنین جستجوی هدف، به‌طور طبیعی توسط ملخ انجام می‌گیرد. مدل ریاضی که برای شبیه‌سازی رفتار ملخ‌ها استفاده می‌شود، فرمول (۱) می‌باشد [۲۷]

$$X_i = S_i + G_i + A_i \quad (1)$$

در فرمول X_i موقعیت i امین ملخ را مشخص می‌کند، S_i تعامل اجتماعی و G_i نیروی گرانش اعمال شده به ملخ i ام می‌باشد، A_i جهت باد را نمایش می‌دهد. برای ایجاد رفتار تصادفی می‌توان رابطه (۱) را به صورت زیر بازنویسی کرد؛ که در فرمول (۲) r_1, r_2, r_3 اعداد تصادفی در بازه $[0, 1]$ می‌باشند.

$$X_i = r_1 S_i + r_2 G_i + r_3 A_i \quad (2)$$

مقدار S_i یعنی تعامل اجتماعی برای ملخ i ام با توجه به فرمول (۳) محاسبه می‌شود.

$$S_i = \sum_{j \neq i}^N s(d_{ij}) \hat{d}_{ij} \quad (3)$$

که در آن d_{ij} فاصله بین ملخ i ام تا ملخ j ام را نشان می‌دهد به صورت $d_{ij} = |x_j - x_i|$ محاسبه می‌شود. s یک تابع برای تعریف فشار نیروی اجتماعی می‌باشد همان‌طور که در رابطه (۳) نشان داده شده است. $\hat{d}_{ij} = \frac{x_j - x_i}{d_{ij}}$ یک بردار واحد از i امین ملخ به j امین ملخ می‌باشد.

هر دیتاست شامل چندین ماژول نرم‌افزاری است که متریک‌های کیفیت به‌عنوان ورودی ماژول‌ها می‌باشد. هر ماژول شامل یک برچسب خروجی، از خطادار بودن یا بدون خطاست که نشان می‌دهد. خطایی در ماژول مربوطه دیده شده است. برچسب‌گذاری ماژول‌ها به‌صورت دستی بعد از مرحله آزمون انجام می‌شود [۱]. شرح مختصری از ویژگی‌های مجموعه داده‌هایی که در این تحقیق مورد استفاده قرار گرفته‌اند در جدول ۱ ذکر شده است.

در ستون اول این جدول نام مجموعه داده‌ها آورده شده و در ستون‌های بعدی به ترتیب زبان برنامه‌نویسی، تعداد ویژگی‌ها، تعداد ماژول‌ها و درصد خطای موجود در هر مجموعه داده که نمایانگر درصد ماژول‌های خطادار است نشان داده شده است.

جدول ۱: ویژگی‌های مجموعه داده استفاده شده

| مجموعه داده | زبان برنامه‌نویسی | تعداد ویژگی | تعداد ماژول | درصد خطا |
|-------------|-------------------|-------------|-------------|----------|
| Cm1 | C | ۲۱ | ۴۹۸ | ٪۱۰ |
| Jm1 | C | ۲۱ | ۱۰۸۸۵ | ٪۱۹ |
| Kc1 | C++ | ۲۱ | ۲۱۰۹ | ٪۱۵ |
| Kc2 | C++ | ۲۱ | ۵۲۲ | ٪۲۱ |
| Kc3 | Java | ۳۹ | ۴۲۹ | ٪۹ |
| Pc1 | C | ۲۱ | ۱۱۰۹ | ٪۷ |
| Pc2 | C | ۳۷ | ۴۵۰۵ | ٪۰٫۵ |
| Pc3 | C | ۳۷ | ۱۵۶۳ | ٪۱۰ |
| Pc4 | C | ۳۷ | ۱۴۵۸ | ٪۱۲ |

دیتاست ناسا شامل ۲۱ متریک از نوع مکایب و هالستد، مشتق شده هالستد است. تعدادی از محققین بر این نظرند که متریک‌های مشتق شده هالستد برای پیش‌بینی نقص نرم‌افزار زیاد کاربردی ندارند. به همین دلیل کمتر استفاده می‌شوند. متریک‌های مکایب و هالستد به‌طور فراوان برای اندازه‌گیری کیفیت ماژول‌های نرم‌افزار به کار می‌روند. هر ماژول به‌واسطه زیر برنامه یا متد یا تابع در زبان‌های برنامه‌نویسی روالی مانند C و به‌وسیله کلاس در زبان‌های شی گرا مانند (C++, java) نمایش داده می‌شود [۱]. جدول ۲ خلاصه‌ای از ۲۱ متریک که برای هر ماژول

می‌باشد. C ثابت کاهش برای کم کردن منطقه آسایش، دافعه و جاذبه می‌باشد. S تقریباً مشابه مؤلفه S در رابطه (۱) می‌باشد. با این حال ما گرانش را در نظر نمی‌گیریم (مؤلفه G) و فرض می‌کنیم که جهت باد (مؤلفه A) همیشه به سمت هدف می‌باشد.

در رابطه (۸) اولین C از سمت چپ، این جابجایی‌های ملخ‌ها در اطراف هدف را کاهش می‌دهد. به عبارت دیگر، این پارامتر تعادل بین اکتشاف و بهره‌برداری در اطراف هدف را ایجاد می‌کند. متغیر C دوم ناحیه جاذبه، دافعه و آسایش بین ملخ‌ها را کاهش می‌دهد. مؤلفه $s(|x_j^d - x_i^d|)$ در رابطه (۸) در نظر گیرید که در آن $C \frac{ub_d - lb_d}{2}$ به‌صورت خطی فضای بین ملخ‌هایی که باید اکتشاف و بهره‌برداری انجام دهند را کاهش می‌دهد. مؤلفه $s(|x_j^d - x_i^d|)$ نشان می‌دهد اگر یک ملخ باید از هدف دفع (اکتشاف) شود یا به هدف (بهره‌برداری) جذب شود.

ضریب C منطقه آسایش متناسب با تعداد تکرارها را کاهش می‌دهد. به‌صورت فرمول (۹) محاسبه می‌شود:

$$C = C_{max} - l \frac{C_{max} - C_{min}}{L} \quad (9)$$

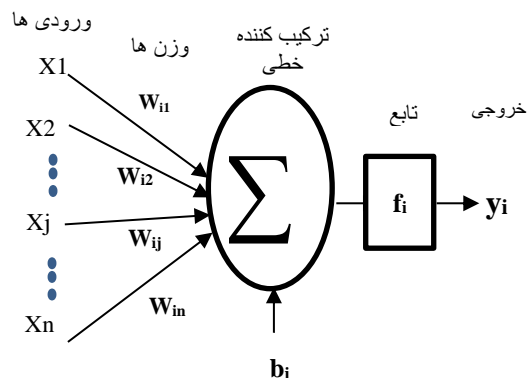
در رابطه (۹) بیشترین مقدار و کمترین مقدار را دارد. L شماره تکرار فعلی را نشان می‌دهد. L حداکثر تعداد تکرار الگوریتم است. پارامتر C تأثیر زیادی بر روی حرکت ملخ‌ها و همگرایی آن‌ها دارد. الگوریتم GOA^۱ بهینه‌سازی را با ایجاد یک مجموعه تصادفی از راه‌حل‌ها شروع می‌کند. عامل‌های جستجو موقعیت خود را بر اساس رابطه (۸) به‌روزرسانی می‌کنند. موقعیت بهترین هدف به‌دست‌آمده تاکنون در هر تکرار به‌روزرسانی می‌شود. علاوه بر این، فاکتور C با رابطه (۹) محاسبه می‌شود و فاصله بین ملخ‌ها به بازه [۱،۴] در هر تکرار نرمال می‌شود. به‌روزرسانی موقعیت تا زمانی که شرایط خاتمه برقرار نشود تکرار می‌شود. موقعیت و شایستگی بهترین هدف در نهایت به‌عنوان بهترین تقریب از بهینه سراسری به خروجی داده می‌شود.

۴- توصیف مخزن داده

دیتاست ناسا معمولاً به‌عنوان دیتاست معیار سنجش برای مسائل پیش‌بینی خطا به کار می‌رود. پروژه‌های نرم‌افزاری

^۱ Grasshopper Optimization Algorithm

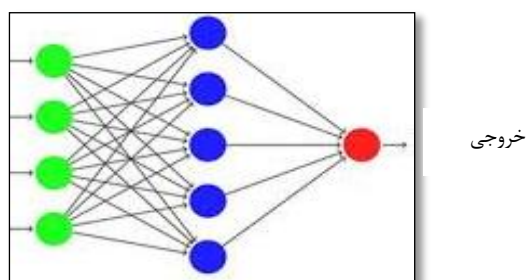
است. به دو نوع شبکه‌های پیش‌خور (FFN^۱) و پس‌خور (RNN^۲) تقسیم می‌شوند. تفاوت آن‌ها در این است که در شبکه‌های پس‌خور، حداقل یک سیگنال برگشتی از یک نرون به همان نرون یا نرون‌های همان لایه و یا لایه قبل وجود دارد. اغلب ۸۰ درصد از کاربردها از شبکه‌های عصبی پیش‌خور استفاده می‌شود.



شکل ۱: شکل کلی یک پرسپترون [۱]

شکل (۱) یک پرسپترون با اجزای نشان داده شده است. شبکه عصبی پرسپترون از چند پرسپترون و چندلایه ایجاد می‌شود. در شبکه‌های پرسپترون چندلایه هر نرون در هر لایه به تمام نرون‌های لایه قبل متصل می‌باشد. به چنین شبکه‌هایی، شبکه‌های کاملاً مرتبط می‌گویند. جهت آموزش شبکه‌های پرسپترون، الگوریتم‌های آموزشی مختلفی وجود دارد. شکل (۲) ساختار یک MLP را با دولایه مخفی و یک لایه ورودی و یک لایه خروجی نشان می‌دهد.

لایه خروجی لایه پنهان لایه ورودی



شکل ۲: ساختار شبکه عصبی mlp

خروجی شبکه عصبی طبق فرمول (۱۰) به دست می‌آید:

$$Y_i = f_i(\sum_{j=1}^n x_j w_{ij} + b_i) \quad (10)$$

که Y_i خروجی شبکه و x_i ورودی شبکه می‌باشند. w_{ij}

جمع‌آوری شده است را نشان می‌دهد.

جدول ۲: خلاصه‌ای از متریک‌ها (متد-سطح) [۱]

| نوع | متریک | شرح | |
|-------------|------------------|---------------------------------|----------------------|
| مکایب | loc | تعداد خطوط کد | |
| | V(g) | تعداد مسیرهای مستقل e- n+2 | |
| | Ev(g) | پیچیدگی EV(G)=V(G)-m | |
| | Iv(g) | میزان پیچیدگی | |
| هالستد | n | تعداد کل عملگرها و عملوندها | |
| | v | تعداد مقایسه‌های N*log2(mu) | |
| | l | میزان هوشمندی 'L*V' | |
| | d | میزان سختی | |
| | i | میزان هوشمندی | |
| | e | میزان تخمین خطا | |
| | b | زمان مصرفی E/18 و برحسب ثانیه) | |
| | Locoed | تعداد خطوط کد | |
| | IOComment | تعداد خطوط توضیحات | |
| | IOBlank | تعداد خطوط خالی. | |
| | IOCodeAndComment | تعداد خطوط شامل کد و توضیحات. | |
| | مشتق شده هالستد | uniqOp | تعداد عملگرهای یکتا |
| | | uniqOpnd | تعداد عملوندهای یکتا |
| totalOp | | تعداد کل عملگرها | |
| totalOpnd | | تعداد کل عملوندها | |
| branchCount | | تعداد کل | |

۵- شبکه عصبی

امروزه شبکه‌های عصبی کاربرد فراوانی در پیش‌بینی خطا دارند [۲۸-۳۰]. شبکه عصبی مدلی بر اساس مغز انسان

² Recurrent Neural Network

¹ Feed Forward Network

۶-مدل پیشنهادی

شبکه عصبی یکی از روش‌های طبقه‌بندی دقیق و تقریب زنده‌های جهانی برای پیش‌بینی نقص نرم‌افزار است. برای آموزش وزن‌های شبکه عصبی، از یک شبکه عصبی با ساختار مشخص و معینی، یعنی تعداد ثابتی لایه پنهانی و نرون در هر لایه برای شبکه موردنظر استفاده می‌کنیم. فرآیند آموزش وزن‌های شبکه با همان معماری مشخص آغاز شده و تا انتهای آموزش همان ساختار را برای شبکه داشتیم. مشخص نیست که آن ساختار بهترین ساختار موجود برای شبکه عصبی است. به‌روزرسانی ساختار شبکه عصبی تبدیل به یک مسئله بهینه می‌شود. از جهتی الگوریتم ملخ، یک روش هوش جمعی و قوی و جدید برای یافتن نقاط بهینه سراسری در فضای جستجو است. بنابراین می‌توان از الگوریتم ملخ، برای پیدا کردن ساختار بهینه در شبکه عصبی استفاده کرد. در روش پیشنهادی برای پیش‌بینی نقص‌های موجود در نرم‌افزار، یک سیستم ترکیبی از شبکه‌های عصبی و الگوریتم ملخ استفاده شده است. در آن برای بالا بردن دقت و صحت سیستم و فرار از به دام افتادن الگوریتم آموزشی شبکه عصبی در مینیمم محلی، ابتدا الگوریتم ملخ وزن‌های بهینه‌شده را که به دقت قابل قبولی رسیده است، را به الگوریتم آموزشی پس‌انتشار خطا منتقل می‌کند.

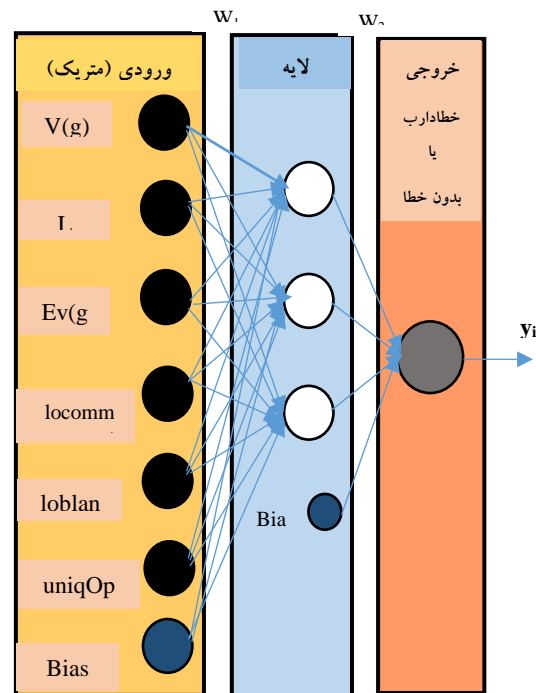
به‌عبارت‌دیگر، در روش پیشنهادی تمرکز بر این است که نقطه شروع الگوریتم آموزشی شبکه عصبی که الگوریتم پس‌انتشار خطا می‌باشد، به‌گونه‌ای انتخاب کنیم که در مینیمم‌های محلی گیر نکند. اگر الگوریتم پس‌انتشار با یک شرایط اولیه خوب و نزدیک وزن بهینه عمل جستجو را شروع کند، با سرعت بالایی به نقطه بهینه نزدیک می‌شود. در الگوریتم ترکیبی ملخ و پس‌انتشار خطا، در مراحل اولیه، پس از به دست آوردن بهترین ساختار شبکه عصبی توسط الگوریتم ملخ، وزن‌های شبکه عصبی نیز توسط این الگوریتم پیدا می‌شوند. پس از یک تعداد تکرار معین یا پس از رسیدن به یک دقت از پیش تعیین‌شده، الگوریتم یادگیری از الگوریتم ملخ به الگوریتم پس‌انتشار خطا سوییچ می‌شود. به‌این ترتیب الگوریتم پس‌انتشار خطا با یک شرایط اولیه خوب و در نزدیکی نقطه بهینه سراسری شروع می‌شود. و طی تکرارهای کم به دقت بالایی می‌رسد. بنابراین در مدل پیشنهادی، علاوه بر بهینه کردن وزن‌های شبکه عصبی، ساختار آن را نیز توسط الگوریتم ملخ بهینه

وزن‌های اتصالی بین گره‌های ورودی و خروجی است. b_i بایاس می‌باشد. f_i تابع انتقال است. الگوریتم یادگیری پس‌انتشار خطا، یکی از رایج‌ترین الگوریتم‌ها جهت آموزش شبکه‌های عصبی MLP می‌باشد. به عبارتی توپولوژی شبکه‌های MLP، با قانون یادگیری پس‌انتشار خطا تکمیل می‌شود. این قانون تقریبی از الگوریتم بیشترین نزول [۳۱] است. به‌طور خلاصه، فرآیند پس‌انتشار خطا از دو مسیر اصلی تشکیل می‌شود. مسیر رفت و مسیر برگشت. در مسیر رفت، یک الگوی آموزشی به شبکه اعمال می‌شود و تأثیرات آن از طریق لایه‌های میانی به لایه خروجی انتشار می‌یابد. تا اینکه نهایتاً خروجی واقعی شبکه MLP، به دست می‌آید. در این مسیر، پارامترهای شبکه (ماتریس‌های وزن و بردارهای بایاس)، ثابت و بدون تغییر در نظر گرفته می‌شوند. در مسیر برگشت، برعکس مسیر رفت، پارامترهای شبکه MLP تغییر و تنظیم می‌گردند. این تنظیمات بر اساس قانون یادگیری اصلاح خطا انجام می‌گیرد. سیگنال خطا، در لایه خروجی شبکه تشکیل می‌گردد. [۳۲]

تابع خطای E به صورت (۱۱) می‌باشد:

$$E(W(t)) = \frac{1}{n} \sum_{j=1}^n \sum_{k=1}^n (dk - ok) \quad (11)$$

برای درک بهتر نقش شبکه عصبی در مدل پیشنهادی، ساختار شبکه عصبی برای مجموعه داده pc1 در شکل (۳) نشان داده شده است.



شکل ۳: ساختار شبکه عصبی برای دیتاست pc1 [۱]

می‌کنیم. در واقع یک الگوریتم ملخ در بدنه سیستم پیشنهادی داریم، که فرآیند بهینه‌سازی ساختار شبکه عصبی را انجام می‌دهد. یک جمعیت از ملخ‌ها در داخل این بدنه داریم که فرآیند بهینه‌سازی وزن‌های شبکه عصبی را انجام می‌دهد.

۱-۶- مرحله آموزش در روش پیشنهادی

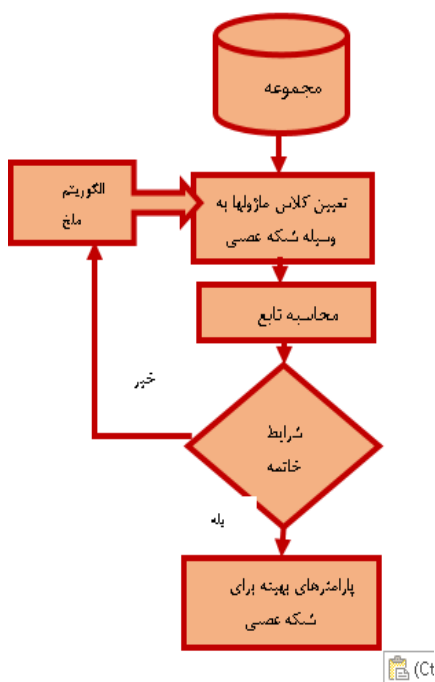
با استفاده از ارزیابی متقاطع نمونه اصلی دیتاست به ده زیر نمونه تصادفی تقسیم می‌شود. از این ۱۰ تا زیر نمونه، ۹ تای آن برای داده‌های آموزشی استفاده می‌شود و ۱ زیرنمونه، برای تست مدل استفاده می‌شود. علت تقسیم دیتاست به دو تا این است که شبکه عصبی فقط با یکسری داده‌های خاص آموزش نبیند. اگر داده‌های جدیدی به‌عنوان ورودی به آن داده شود خوب عمل کند. شبکه عصبی قابلیت تعمیم دهی داشته باشد و خطای *overfitting* روی ندهد. سپس الگوریتم بهینه‌ساز ملخ فراخوانی می‌شود، در هر مرحله وزن‌های شبکه عصبی را بهینه می‌کند. داده‌های آموزشی و تعداد نرون‌ها به ورودی الگوریتم ملخ داده می‌شود. توسط تابع هدف الگوریتم ملخ، وزن‌های W_{e1}, W_{e2} و بایاس‌های b_{i1}, b_{i2} و خطای شبکه عصبی محاسبه می‌شود و به‌عنوان خروجی الگوریتم به دست می‌آید. مراحل آموزش شبکه عصبی توسط الگوریتم ملخ در شکل (۴) نشان داده شده است. این شکل نمایانگر مراحل آموزش شبکه عصبی برای پیش‌بینی می‌باشد.

۲-۶- تابع شایستگی در روش پیشنهادی

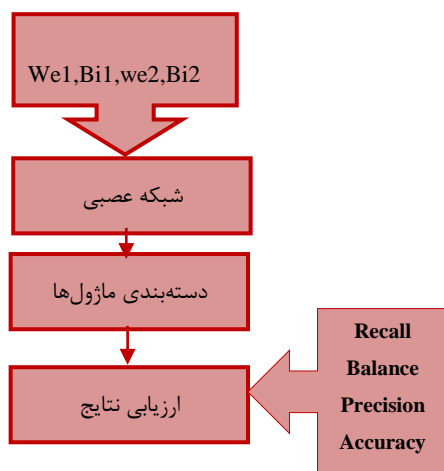
در مدل پیشنهادی، بعد از ایجاد ساختار شبکه عصبی و مشخص کردن وزن‌ها، بایاس‌ها و الگوریتم آموزشی پس‌انتشار خطا به کمک تابع $mse()$ (میانگین مربعات خطا) هزینه محاسبه می‌شود. این تابع مقدار موردنظر (*triny*) را از خروجی شبکه عصبی کم می‌کند و بهترین حالت این است که خروجی تابع $mse()$ صفر شود.

۳-۶- مرحله آزمون در روش پیشنهادی

پس از انجام مرحله آموزش شبکه عصبی در مدل پیشنهادی، وارد مرحله آزمون و تست مدل پیشنهادی جهت سنجش عملکرد شبکه عصبی در مدل پیشنهادی می‌شود. نحوه انجام مرحله آزمون شبکه عصبی در شکل (۵) نشان داده شده است. پس از انجام آموزش مدل پیشنهادی، یکسری پارامتر از مدل آموزش به دست می‌آید.



شکل ۴: آموزش شبکه عصبی در مدل پیشنهادی



شکل ۵: آزمون شبکه عصبی در مدل پیشنهادی

به‌عنوان ورودی در مرحله آزمون از آن‌ها استفاده می‌شود. علاوه بر پارامترهای $W_{e1}, W_{e2}, b_{i1}, b_{i2}$ دو پارامتر X و Y نیز وجود دارد که این پارامترها مربوط به داده‌های تست می‌باشد. اکنون باید شبکه عصبی با پارامترهای بهینه‌شده ماژول‌های تست را مورد پیش‌بینی قرار دهد. در این بخش از روش ارزیابی متقاطع که در آن نمونه اصلی به‌صورت تصادفی به ۱۰ زیرنمونه تقسیم می‌شود، استفاده شده است. از ۱۰ زیرنمونه ۹ تای آن‌ها برای داده‌های آموزشی و یک زیرنمونه به‌منظور تست مدل استفاده می‌شود. همچنین فرآیند ارزیابی متقاطع، ۱۰ بار تکرار

جدول ۴: پارامترهای الگوریتم ملخ در مدل پیشنهادی

| پارامتر | مقدار |
|----------------------------------|---|
| SearchAgents_no | تعداد عامل‌های جستجو |
| Max_iter | تعداد تکرار |
| dim | تعداد پارامترهای مسئله که با توجه به تعداد نرون‌ها محاسبه می‌شود. |
| $w_{e1}, w_{e2}, b_{i1}, b_{i2}$ | وزن‌ها و بایاس‌های بهینه‌شده. |

جدول ۵: پارامترهای شبکه عصبی در مدل پیشنهادی

| پارامتر | مقدار |
|--|-----------------------------------|
| neu | تعداد نرون‌های لایه‌های شبکه عصبی |
| $w_{e1}, w_{e2}, b_{i1}, b_{i2}, tsX, tsY$ | پارامترهای ورودی شبکه عصبی |
| err | محاسبه خطای شبکه عصبی |

۶-۶- نتایج تجربی در مدل پیشنهادی

در این بخش نتایج حاصل از اجرای روش‌های پیشنهادی بر روی ۹ پایگاه داده مختلف نمایش داده شده است. این نتایج مربوط به داده‌های تست می‌باشد. با اعمال ارزیابی متقاطع ۱۰ تایی به دست آمده است. جدول ۶ نتایج حاصل از اجرای روش پیشنهادی بر روی پایگاه داده‌های مختلفی نشان می‌دهد:

جدول ۶: نتایج مدل پیشنهادی روی ۹ دیتاست

| مخزن داده | توازن | فراخوانی | دقت | صحت |
|-----------|--------|----------|--------|--------|
| Cm1 | ۰,۹۸۵۰ | ۰,۹۶۰۰ | ۰,۹۷۰۰ | ۰,۹۲۳۰ |
| Kc1 | ۰,۹۸۰۰ | ۰,۹۸۵۰ | ۰,۸۷۲۰ | ۰,۹۴۲۰ |
| Kc2 | ۰,۹۸۴۶ | ۰,۹۰۰۰ | ۰,۸۰۰۰ | ۰,۹۲۰۰ |
| Kc3 | ۰,۹۸۴۶ | ۰,۹۵۰۰ | ۰,۸۸۲۰ | ۰,۹۶۰۰ |
| Jm1 | ۰,۹۸۳۹ | ۰,۹۸۰۰ | ۰,۸۸۷۱ | ۰,۹۸۱۴ |
| Pc1 | ۰,۹۸۴۳ | ۰,۹۸۱۲ | ۰,۹۰۳۰ | ۰,۹۸۱۸ |
| Pc2 | ۰,۸۲۳۲ | ۰,۹۷۴۲ | ۰,۹۹ | ۰,۹۸۸۸ |
| Pc3 | ۰,۹۸۳۹ | ۰,۹۷۱۵ | ۰,۸۵۷۱ | ۰,۹۸۱۱ |
| Pc4 | ۰,۸۵۸۶ | ۰,۹۷۲۰ | ۰,۹۷۰۰ | ۰,۹۳۲۰ |

می‌شود. میانگین نتایج حاصل از ۱۰ بار اجرا به عنوان نتایج نهایی مورد استفاده قرار می‌گیرد. X ماتریسی $m \times n$ و Y ماتریسی $1 \times n$ می‌باشد. m معرف تعداد ویژگی‌ها و n تعداد ماژول‌ها در هر مجموعه داده نشان می‌دهد. ارزیابی متقاطع موجب می‌شود دقت آزمون مدل بالا برود.

۶-۴- پارامترهای ارزیابی مدل پیشنهادی

کارایی مدل پیشنهادی با استفاده از معیارهای ارزیابی، به کمک ماتریس درهم‌ریختگی در جدول (۳) نشان داده شده است. این ماتریس نمونه‌هایی که برای هر کلاس به درستی و به اشتباه تشخیص داده شده‌اند، را نمایش می‌دهد. TP و TN به ترتیب تعداد نمونه‌های کلاس مثبت و منفی که به درستی طبقه‌بندی شده‌اند را نشان می‌دهند. FN تعداد نمونه‌های کلاس مثبت می‌باشد که به اشتباه منفی پیش‌بینی شده‌اند. در حالی که FP تعداد نمونه‌های منفی می‌باشد که به اشتباه مثبت معرفی شده‌اند. مورد بررسی قرار گرفت.

جدول ۳: ماتریس درهم‌ریختگی

| دسته‌بندی پیش‌بینی کننده | دسته‌بندی واقعی | |
|--------------------------|-----------------|-------------|
| | حاوی نقص | عاری از نقص |
| حاوی نقص | TP | FP |
| عاری از نقص | FN | TN |

در این مقاله کارایی مدل پیشنهادی با استفاده از مشهورترین و پرکاربردترین معیارها شامل موارد فراخوانی مجدد (۱۲)، توازن (۱۳)، دقت (۱۴)، صحت (۱۵) می‌باشد اندازه‌گیری شده است:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (12)$$

$$B = 1 - \sqrt{\frac{1}{2} \left(\left(\frac{FN}{TP + FN} \right)^2 + \left(\frac{FP}{TN + FP} \right)^2 \right)} \quad (13)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (14)$$

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN} \quad (15)$$

۶-۵- پارامترهای تنظیم شده مدل پیشنهادی

پارامترهای مؤثر در الگوریتم ملخ و شبکه عصبی در جدول ۴ و ۵ آمده است. این مقادیر با سعی و خطا به صورت تجربی تعیین شده است. معیار انتخاب مقادیر، رسیدن به نتایج بهینه‌تر بوده است.

۷-ارزیابی نتایج

برای نمایش عملکرد و کارآمدی مدل پیشنهادی، نتایج به دست‌آمده را با شش روش متداول درخت تصمیم j48^۱، نایو بیس^۲، جنگل تصادفی^۳ [۳۳]، الگوریتم ترکیبی شبکه عصبی حساس به هزینه و زنبورعسل [۱]، شبکه عصبی شعاعی [۳۰] و الگوریتم ترکیبی شبکه عصبی و جهش قورباغه [۳۴] مقایسه نمودیم. به منظور درک بهتر، نتایج کلی در جداول ۷ تا ۱۰، در مقابل با سایر روش‌ها نمایش داده شده و نمودار ستونی نتایج نیز رسم شده است.

با توجه به جدول (۷)، مدل پیشنهادی در همه دیتاست‌ها در رتبه اول قرار دارد. الگوریتم شبکه عصبی در دیتاست‌های (kc1,cm1,kc3,pc1) در رتبه دوم قرار دارد. الگوریتم RF در دیتاست های (kc1,cm1,pc1,pc2,pc3,pc4) در رتبه سوم و الگوریتم j48 در رتبه چهارم قرار دارد. پایین‌ترین رتبه تقریباً در همه دیتاست‌ها متعلق به الگوریتم نایو بیس و در دیتاست (kc1,cm1) متعلق به شبکه عصبی حساس به هزینه است. نمودار مربوطه در شکل شماره (۶) بخش پیوست درج شده است.

جدول ۷: ارزیابی مقدار صحت در مقابل با سایر روش‌ها

| دیتاست الگوریتم | pc4 | pc3 | pc2 | pc1 | kc3 | cm1 | kc1 |
|-----------------|-----|------|-----|------|-------|-------|-------|
| j48 | ۸۹ | ۸۴ | ۹۸ | ۸۹ | ۷۹ | ۸۳ | ۸۱ |
| NB1 | ۸۵ | ۸۴ | ۹۸ | ۸۹ | ۷۷ | ۸۳ | ۸۱ |
| RF | ۸۹ | ۸۵ | ۹۸ | ۸۸ | ۷۵ | ۸۲ | ۸۳ |
| sfl | - | - | - | - | - | ۷۹,۴۵ | ۷۸,۵۳ |
| Mlp+ | - | - | - | - | - | ۳۴,۱۷ | ۳۳,۶۷ |
| mlp | - | - | - | ۹۴,۲ | ۹۴,۷۴ | ۹۰,۷۵ | ۸۶,۷۵ |
| مدل پیشنهادی | ۹۷ | ۸۵,۷ | ۹۹ | ۹۰,۳ | ۸۸,۲ | ۹۷ | ۸۷,۲ |

جدول شماره ۸، مقدار معیار دقت در مقایسه با سایر روش‌ها را نشان می‌دهد. مدل پیشنهادی در دیتاست های (pc4,pc3,pc2,cm1,kc1) نتایج بهبودیافته‌ای به دست آورده است. در این دیتاست‌ها رتبه یک را دارد. رتبه ۲ را الگوریتم شبکه عصبی در دیتاست های (kc1,cm1,kc3,pc1) دارد. رتبه ۳ را الگوریتم جنگل

تصادفی در دیتاست های (kc1,pc3) دارد. در مابقی دیتاست‌ها رتبه الگوریتم نایوی بیس و درخت جنگلی و الگوریتم جنگل تصادفی یکسان می‌باشد. پایین‌ترین رتبه را الگوریتم ترکیبی شبکه عصبی حساس به هزینه در دیتاست (cm1,kc1) دارد. نمودار مربوطه در شکل شماره (۷) بخش پیوست درج شده است.

جدول ۸: ارزیابی مقدار دقت در مقابل با سایر روش‌ها

| دیتاست الگوریتم | pc4 | pc3 | pc2 | pc1 | kc3 | cm1 | kc1 |
|-----------------|-----|------|-----|------|-------|-------|-------|
| j48 | ۸۹ | ۸۴ | ۹۸ | ۸۹ | ۷۹ | ۸۳ | ۸۱ |
| NB1 | ۸۵ | ۸۴ | ۹۸ | ۸۹ | ۷۷ | ۸۳ | ۸۱ |
| RF | ۸۹ | ۸۵ | ۹۸ | ۸۸ | ۷۵ | ۸۲ | ۸۳ |
| sfl | - | - | - | - | - | ۷۹,۴۵ | ۷۸,۵ |
| Mlp+ | - | - | - | - | - | ۳۴,۱۷ | ۳۳,۶۷ |
| mlp | - | - | - | ۹۴,۲ | ۹۴,۷۴ | ۹۰,۷۵ | ۸۶,۷ |
| مدل پیشنهادی | ۹۷ | ۸۵,۷ | ۹۹ | ۹۰,۳ | ۸۸,۲ | ۹۷ | ۸۷,۲ |

جدول شماره ۹، ارزیابی مقدار معیار فراخوانی در مقایسه با سایر روش‌ها را نمایش می‌دهد. مدل پیشنهادی مقدار فراخوانی را بهبود داده است. در مقایسه با الگوریتم ترکیبی شبکه عصبی حساس به هزینه رتبه یک را دارد.

جدول ۹: ارزیابی مقدار فراخوانی در مقابل با سایر روش‌ها

| دیتاست الگوریتم | pc1 | jm1 | kc3 | kc2 | kc1 |
|-----------------|-------|-------|-------|-------|-------|
| Mlp+ | ۷۹,۹۳ | ۸۰,۲۶ | ۷۹,۹۳ | ۷۲,۲۲ | ۹۱,۳۱ |
| مدل پیشنهادی | ۹۰,۳ | ۹۳,۲ | ۹۸ | ۹۰ | ۹۶ |

شکل شماره (۸)، نمودار مقدار فراخوانی در مقایسه با سایر روشها را نشان می‌دهد.

جدول شماره ۱۰، ارزیابی مقدار معیار توازن در مقایسه با سایر روش‌ها را نمایش می‌دهد. مدل پیشنهادی مقدار توازن را بهبود داده است و در مقایسه با الگوریتم ترکیبی شبکه عصبی حساس به هزینه رتبه یک را دارد. شکل شماره (۹)، نمودار مقدار توازن در مقایسه با سایر روشها را نشان می‌دهد.

³ Random Forest

¹ Decision tree J48

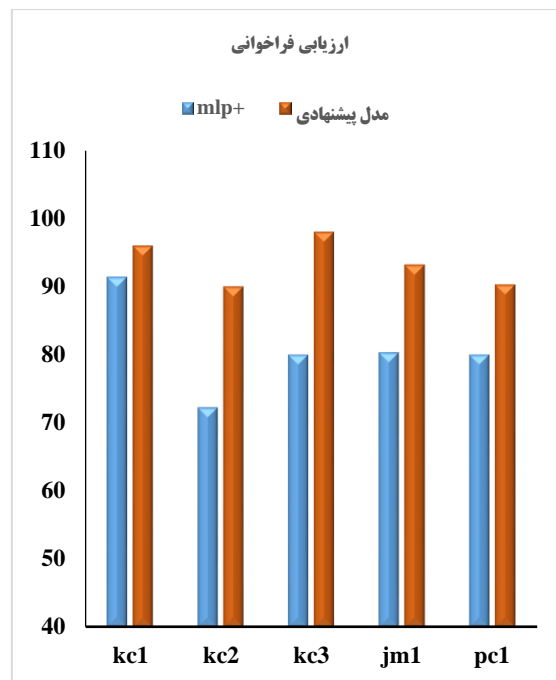
² Naive Bayes

جدول ۱۱: ارزیابی زمان اجرا در مقابل با سایر روش‌ها

| دیتاست الگوریتم | pc1 | jm1 | kc3 | kc2 | kc1 |
|--------------------|-----|-----|-----|-----|-----|
| j48 | ۲۳ | - | ۵۸ | ۲۰ | ۹۶ |
| NB1 | ۵۹ | ۶۳ | - | ۵۱ | - |
| RF | ۳۱ | ۲۵ | - | ۴۵ | - |
| sfl | ۶۸ | ۶۰ | ۸۶ | - | ۸۳ |
| Mlp+ | ۸۸ | ۵۱ | ۹۳ | - | ۱۰۱ |
| mlp | ۷۰ | ۵۵ | ۸۸ | ۵۱ | ۹۵ |
| مدل پیشنهادی | ۴۹ | ۳۵ | ۶۲ | ۴۵ | ۸۳ |

۸- نتیجه‌گیری و کار آینده

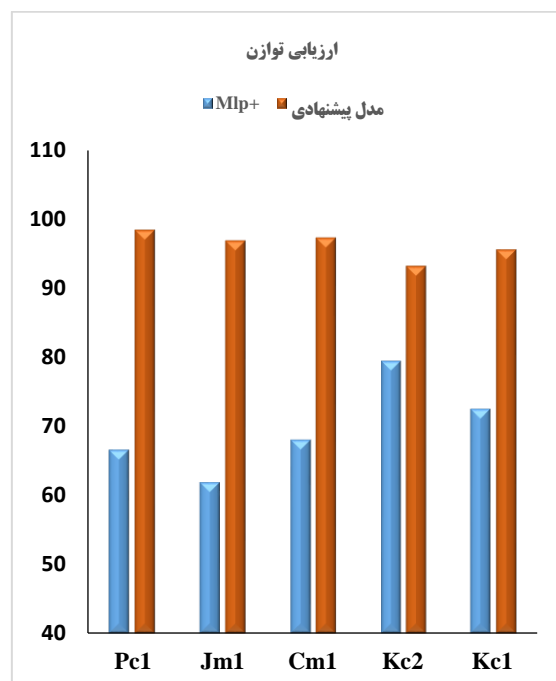
امروزه کیفیت و قابلیت اطمینان نرم‌افزار یکی از مسائل بسیار مهم در زمینه تولید نرم‌افزار می‌باشد. فناوری تشخیص خطا و نقص نرم‌افزار یکی از اهداف تحقیقاتی بسیار مهم در زمینه قابلیت اطمینان سیستم‌های نرم‌افزاری است، که مانع شکست نرم‌افزار می‌شود. از این رو عملکرد مدل پیش‌بینی نقص به منظور پیش‌بینی دقیق نقص در بهبود و اثربخشی مدل‌ها مهم می‌باشد. در این مقاله سعی شده است طبقه‌بندی با تکیه بر مدل‌های مبتنی بر یادگیری ماشین، مدلی ترکیبی و کارآمد به منظور پیش‌بینی نقص نرم‌افزار ارائه شود. مبنای مدل پیشنهادی شبکه عصبی پرسپترون چندلایه است. با کمک الگوریتم ملخ بهبود داده شده تا بتوان با قدرت بهینه‌سازی الگوریتم‌های فراابتکاری و قدرت یادگیری شبکه عصبی ضرایب مناسب و نتایج قابل قبولی تولید کرد. به منظور ارزیابی نتایج مدل پیشنهادی، این مدل را بر روی ۹ مجموعه داده به کار گرفته شده است و مبنای ارائه نتایج، روش ارزیابی متقاطع بوده است. عملکرد مدل پیشنهادی با ۶ مقاله معتبر که جدید منتشر شده است مورد مقایسه قرار گرفت. نتایج نشان داده شده در نمودارها حاکی از آن است که مدل پیشنهادی توانسته است بالاترین مقدار معیارهای ارزیابی صحت، فراخوانی، توازن را در همه مجموعه داده‌ها و معیار ارزیابی دقت در بیشتر مجموعه داده‌ها را به دست آورد. با توجه به ضرورت پیش‌بینی نقص نرم‌افزار در تولید نرم‌افزار مدل پیشنهادی قابل گسترده است برای مثال استفاده از الگوریتم‌های انتخاب ویژگی، ارزیابی مدل پیشنهادی با سایر معیارهای ارزیابی و استفاده از الگوریتم‌های فراابتکاری جدیدتر از دیگر کارهای آینده است.



شکل ۸: ارزیابی مقدار معیار فراخوانی در مقایسه با سایر روش‌ها

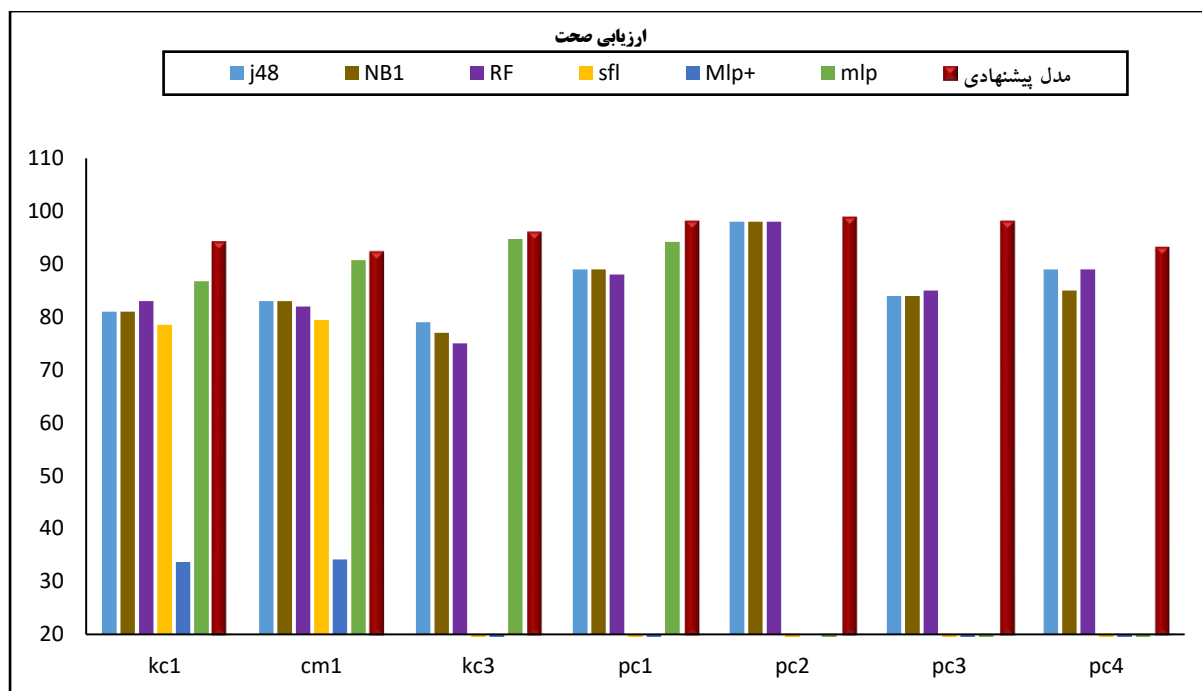
جدول ۱۰: ارزیابی مقدار توازن در مقابل با سایر روش‌ها

| دیتاست الگوریتم | Kc1 | Kc2 | Cm1 | Jm1 | Pc1 |
|--------------------|-------|-------|-------|-------|-------|
| Mlp+ | ۷۲.۵۶ | ۷۹.۴۶ | ۶۸.۰۹ | ۶۱.۹۷ | ۶۶.۶۳ |
| مدل پیشنهادی | ۹۵.۶ | ۹۳.۲ | ۹۷.۳ | ۹۶.۹ | ۹۸.۴۳ |

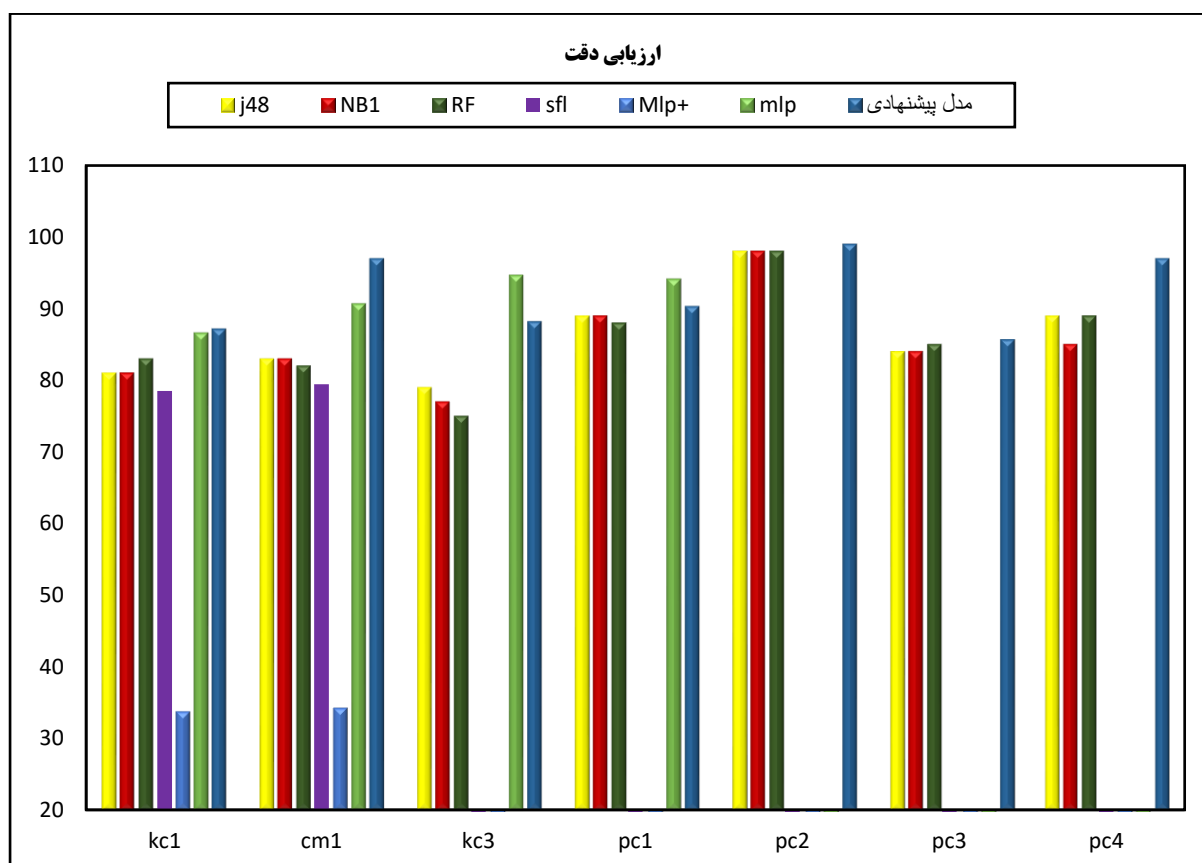


شکل ۹: ارزیابی مقدار معیار توازن در مقایسه با سایر روش‌ها

جدول شماره ۱۱، مقایسه مقدار زمان اجرا مدل پیشنهادی را در مقایسه با سایر روش‌ها نمایش می‌دهد.



شکل ۶: ارزیابی مقدار معیار صحت در مقایسه با سایر روش‌ها



شکل ۷: ارزیابی مقدار معیار دقت در مقایسه با سایر روش‌ها

مراجع

- [1] Ö. F. Arar, and K. Ayan, "Software defect prediction using cost-sensitive neural network". Applied Soft Computing, Vol. 33, August 2015, pp. 263-277.
- [2] J. Cahill, J. M. Hogan, and R. Thomas, "Predicting fault-prone software modules with rank sum classification". in Software Engineering Conference (ASWEC), 2013 22nd Australian, IEEE, (2013)
- [3] W. Han, H. Jiang, W. Li, and Y. Li, "A summary of software defect model. in Control and Automation (CA)", 2014 7th Conference on, IEEE, (2014)
- [۴] مهسا عاشمی مجد و عباس رسولزادگان و زهرا قویدل یزدی "مروری نظام‌مند بر مدل‌سازی قابلیت اطمینان نرم‌افزار". مدل‌سازی در مهندسی، دوره ۱۵، شماره ۵۰، پاییز ۱۳۹۶، صفحه ۳۱۳-۲۸۵.
- [5] T. M. Khoshgoftaar, and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study". Empirical Software Engineering, Vol. 9, No.3, September 2004, pp. 229-257.
- [6] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An Empirical Study on Software Defect Prediction with a Simplified Metric Set", vol. 59. Wuhan University, Wuhan, March 2014, pp. 170-190.
- [7] G. Czubala, Z. Marian, and I. G. Czubala, "Software defect prediction using relational association rule mining". Information Sciences, Vol. 264, April 2014, pp. 260-278.
- [8] R. Malhotra, and A. Jain, "Fault prediction using statistical and machine learning methods for improving software quality", Journal of Information Processing Systems, Vol. 8, No.2, 2012, pp. 241-262.
- [9] I. Gondra, "Applying machine learning to software fault-proneness prediction", Journal of Systems and Software, Vol. 81, No.2, February 2008, pp. 186-195.
- [10] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward comprehensible software fault prediction models using bayesian network classifiers", IEEE Transactions on Software Engineering, Vol. 39, No.2, February 2013, pp. 237-257.
- [11] H. Lu, B. Cukic, and M. Culp, "Software defect prediction using semi-supervised learning with dimension reduction", Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ACM, (2012).
- [12] H. Wang, T. M. Khoshgoftaar, and N. Seliya, "How many software metrics should be selected for defect prediction?", Twenty-Fourth International FLAIRS Conference, (2011).
- [13] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction", Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on, IEEE, (2013).
- [14] S. Wang, and X. Yao, "Using class imbalance learning for software defect prediction", IEEE Transactions on Reliability, Vol. 62, No.2, April 2013, pp. 434-443.
- [15] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction", Information and Software Technology, Vol. 54, No.3, April 2012, pp. 248-256.
- [16] A. Paksoy, and M. Göktürk, "Information fusion with dempster-shafer evidence theory for software defect prediction", Procedia Computer Science, Vol. 3, March 2011, pp. 600-605.
- [17] W. Tao, L. Weihua, S. Haobin, and L. Zun, "Software defect prediction based on classifiers ensemble", Journal of Information and Computational Science, Vol. 8, No.16, December 2011, pp. 4241-4254,.
- [18] A. S. Haghighi, M. A. Dezfali, and S. Fakhrahmad, "Applying mining schemes to software fault prediction: A proposed approach aimed at test cost reduction". in Proceedings of the World Congress on Engineering, (2012).
- [19] A. Okutan, and O. T. Yıldız, "Software defect prediction using Bayesian networks", Empirical Software Engineering, Vol. 19, No.1, August 2014, pp. 154-181.
- [20] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction", in Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on, IEEE, (2010).
- [21] P. Zhang, and Y.-t. Chang, "Software fault prediction based on grey neural network", Natural Computation (ICNC), 2012 Eighth International Conference on, IEEE, (2012).
- [22] F. S. Fazel, "A New Method to Predict the Software Fault Using Improved Genetic Algorithm", Bulletin de la Société Royale des Sciences de Liège, Vol. 85, 2016, pp. 187-202.
- [23] K. Sheoran, P. Tomar, and R. Mishra, "Software Quality Prediction Model with the Aid of Advanced Neural Network with HCS", Procedia Computer Science, Vol. 92, 2016, pp. 418-424.

[24] A. Pal, H. Jain, and M. Kumar, "Optimizing Software Error Proneness Prediction Using Bird Mating Algorithm", *Software Project Management for Distributed Computing*, Springer, April 2017, p. 257-287.

[25] X. Rong, F. Li, and Z. Cui, "A model for software defect prediction using support vector machine based on CBA", *International Journal of Intelligent Systems Technologies and Applications*, Vol. 15, No.1, 2016, pp. 19-34.

[26] R. Malhotra and A. Khurana, "Analysis of Evolutionary Algorithms to Improve Software Defect Prediction", (2018).

[27] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application", *Advances in Engineering Software*, Vol. 105, March 2017, pp. 30-47.

[28] R. D. Stutzke, and M. Crosstalk, "Software estimating technology: A survey". Los Alamitos, CA: IEEE Computer Society Press (1997).

[۲۹] سید بهرام بهشتی اول و وحید احمدیان و احسان درویشان، "شناسایی خسارت در سازه با استفاده از پردازش سیگنال و شبکه های عصبی مصنوعی"، نشریه مدل سازی در مهندسی، دوره ۱۶، شماره ۵۲، بهار ۱۳۹۷، صفحه ۲۱-۲۱.

[۳۰] حسین شریف زاده ونیما امجدی، "توزیع بهینه توان راکتیو با استفاده از الگوریتم بهینه‌سازی دسته ذرات"، نشریه مدل سازی در مهندسی، دوره ۴، شماره ۱۸، پاییز ۱۳۸۸، صفحه ۶۷-۷۳.

[31] I. Maleki, A. Ghaffari and M. Masdari, "A new approach for software cost estimation with hybrid genetic algorithm and ant colony optimization", *International Journal of Innovation and Applied Studies*, Vol. 5, No.1, January 2014, pp. 72 .

[32] V. K. Asari, "Training of a feedforward multiple-valued neural network by error backpropagation with a multilevel threshold function", *IEEE Transactions on Neural Networks*, Vol.12, No.6, November 2001, pp.1519-1521.

[33] A. Chug, and S. Dhall, "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm", (2013).

[۳۴] سولماز فرشیدیور و فرشید کی نیا، "پیش بینی نقص نرم افزار با استفاده از ترکیب مدل شبکه های عصبی و الگوریتم فرااکتشافی جهش قورباغه ها"، اولین همایش ملی فناوری اطلاعات و شبکه های کامپیوتری دانشگاه پیام نور، طبس، ۱۳۹۱.