Scientific Journal of Electronic & Cyber Defense
Vol. 12, No. 4, Winter 2025, Serial No. 48 ,pp.107-123
ISSN: 2322-4347, E-ISSN: 2980-8979

A method for software fault prediction with combination of chaos theory and herd horse optimization algorithm

M. Khosravi Farsani*, A. Karimi, S. Khodadadian

*Master's degree, Imam Hossein University, Tehran, Iran

(Received: 2024/07/26, Revised: 2024/12/11, Accepted: 2025/01/02, Published: 2025/02/01)
DOR: https://dor.isc.ac/dor/20.1001.1.23224347.1403.12.4.10.4

Abstract

The existence of an error in a software product causes unexpected performance in the execution of that product. Software error prediction is a process that is done by checking the faulty components of the software before starting its testing process. The fact is that the occurrence of errors in the software cannot be avoided, but the effort is to minimize the number of errors. This work leads to reducing the time and cost of development, increasing customer satisfaction and creating reliability in the software. In this article, a multi-step process is used for software error prediction. In the first stage, the pre-processing operation has been performed to improve the data and then the feature extraction operation has been performed to have a positive effect on the performance of the learning model. The wild horse algorithm was used to extract the feature and the decision tree classification algorithm was used to predict the software error. In order to evaluate the performance of the classification model, the Promise standard data set and evaluation criteria such as accuracy, correctness, recall and F criterion have been used. Finally, a comparison has been made between the proposed method and the method presented in the reference article. The results of the comparison show that the efficiency of the wild horse algorithm (in the proposed method) is better than the Wall algorithm (in paper P) for feature selection. Because the use of chaos theory in the wild horse algorithm to create diversity in the answers improves the performance of the classification model.

Keywords: Software defect, Feature extraction, Wild horse algorithm, Classification, Decision tree algorithm

Cite this article: M. Khosravi Farsani, A. Karimi, S. Khodadadian "Violent behavior detection in surveillance cameras using convolutional and memory neural networks," Electronic and Cyber Defense, vol.12 , no.4 , pp.107-123 , . DOR: https://dor.isc.ac/dor/20.1001.1.23224347.1403.12.4.9.3
 Publisher: Imam Hossein University

*Corresponding Author Email: a.karimi@ihu.ac.ir

"پدافند الکترونیکی و سایبری"
سال دوازدهم، شماره ۴، زمستان ۱۴۰۳، ص ۱۲۳-۱۰۷
شاپا الکترونیکی: 2980-8979 شاپا چاپی: 2322-4347

علمی – پژوهشی

ارائه روشی برای پیش‌بینی خطای نرم‌افزار با ترکیب نظریه آشوب و الگوریتم تکاملی اسب وحشی

محمدرضا خسروی فارسانی، علی کریمی، سعید خدادادیان

۱- کارشناسی ارشد، ۲- استادیار، ۳- کارشناسی ارشد، دانشگاه جامع امام حسین، تهران ایران

(دریافت: ۱۴۰۳/۰۵/۰۵، بازنگری: ۱۴۰۳/۰۹/۲۱، پذیرش: ۱۴۰۳/۱۰/۱۳، انتشار: ۱۴۰۳/۱۱/۱۳)

DOR: https://dor.isc.ac/dor/20.1001.1.23224347.1403.12.4.10.4

چکیده

وجود خطا در یک محصول نرم‌افزاری باعث بروز عملکرد غیرمنتظره در اجرای آن محصول می‌شود. پیش‌بینی خطای نرم‌افزار فرآیندی است که با بررسی مؤلفه‌های معیوب نرم‌افزار قبل از آغاز مراحل آزمون آن انجام می‌شود. واقعیت این است که وقوع خطا در نرم‌افزار قابل‌اجتناب نیست، اما تلاش بر این است که تعداد خطاها را به حداقل برسانیم. این کار منجر به کاهش زمان و هزینه توسعه، افزایش رضایتمندی مشتری و ایجاد قابلیت اطمینان در نرم‌افزار می‌شود. در این مقاله، از یک فرآیند چندمرحله‌ای برای پیش‌بینی خطای نرم‌افزار استفاده‌شده است. در مرحله نخست، عملیات پیش‌پردازش جهت بهبود داده‌ها انجام‌شده و سپس عملیات استخراج ویژگی برای تأثیر مثبت در عملکرد مدل یادگیری انجام‌شده است. برای استخراج ویژگی از الگوریتم اسب وحشی و برای پیش‌بینی خطای نرم‌افزار از الگوریتم طبقه‌بندی درخت تصمیم استفاده‌شده است. به‌منظور ارزیابی عملکرد مدل طبقه‌بندی، از مجموعه داده‌های استاندارد پرامیس و معیارهای ارزیابی ازجمله دقت، صحت، فراخوانی و معیار F استفاده‌شده است. درنهایت، مقایسه‌ای بین روش پیشنهادی و روش ارائه‌شده در مقاله مرجع انجام‌گرفته است. نتایج مقایسه نشان می‌دهد که کارایی الگوریتم اسب وحشی (درروش پیشنهادی) نسبت به الگوریتم وال (در مقاله پایه) برای انتخاب ویژگی بهتر است؛ زیرا استفاده از نظریه آشوب در الگوریتم اسب وحشی جهت ایجاد تنوع در جواب‌ها، باعث بهبود عملکرد مدل طبقه‌بندی‌شده است.

کلیدواژه‌ها: خطای نرم‌افزار، استخراج ویژگی، الگوریتم اسب وحشی، طبقه‌بندی، الگوریتم درخت تصمیم

۱. مقدمه

توسعه نرم‌افزارهای باکیفیت بالا، در حوزه مهندسی نرم‌افزار از اهمیت زیادی برخوردار است. بااین‌وجود، توسعه این‌گونه نرم‌افزارها بسیار پرهزینه است [۱]. امروزه سامانه‌های نرم‌افزاری ازنظر اندازه و پیچیدگی درحال‌رشد هستند، درنتیجه حفظ کیفیت بالای محصول یکی از مهم‌ترین مشکلات پیشروی صنعت نرم‌افزار است [۲]. با توسعه فناوری کامپیوتر، پیچیدگی دستگاه‌های نرم‌افزاری هرروز بیشتر می‌شود به‌این‌ترتیب، شناسایی و کشف خطاهای نرم‌افزاری نیز به‌مراتب مشکل‌تر می‌گردد [۳].

استفاده از مدل‌های پیش‌بینی خطا در نرم‌افزار، ابزارهای مناسب و مقرون‌به‌صرفه‌ای جهت حفظ کیفیت و کاهش خطاهای نرم‌افزار هستند. در حوزه مهندسی نرم‌افزار پیش‌بینی خودکار خطاهای نرم‌افزار از اهمیت زیادی برخوردار است و محققان به‌طور مستمر بر پیش‌بینی‌های دقیق و قابل‌اعتماد در این حوزه تمرکز دارند. تاکنون تحقیقات زیادی در حوزه پیش‌بینی خطاهای نرم‌افزار انجام‌شده است. شیوه کار تمامی این تحقیقات، این‌گونه است که با جمع‌آوری ویژگی‌های نرم‌افزاری برای تمامی ماژول‌ها، یک مدل ریاضی برای پیش‌بینی ماژول‌های مستعد خطا ساخته می‌شود [۴]. به‌طورکلی مدل‌های اصلی پیش‌بینی خطای نرم‌افزار، به سه دسته تقسیم‌شده‌اند [۵]. این مدل‌ها عبارت‌اند از؛ مدل‌هایی که تعداد خطا در ماژول‌ها را پیش‌بینی می‌کنند، مدل‌های دسته‌بندی‌کننده ماژول‌های نرم‌افزاری و مدل‌های طبقه‌بندی کننده ماژول‌های نرم‌افزاری بر اساس میزان احتمال مستعد خطا بودن آن‌ها. اصلی‌ترین مدل پیش‌بینی خطای نرم‌افزار، مدل طبقه‌بندی کننده است که ماژول‌های نرم‌افزاری را در

استناد: خسروی فارسانی، محمدرضا، کریمی، علی، خدادادیان " ارائه روشی برای پیش‌بینی خطای نرم‌افزار با ترکیب نظریه آشوب و الگوریتم تکاملی اسب وحشی"، پدافند الکترونیکی و سایبری، ۱۲(۴)، ۱۲۳-۱۰۷، ۱۴۰۳.
https://dor.isc.ac/dor/20.1001.1.23224347.1403.12.4.10.4

ناشر: دانشگاه جام امام حسین(ع).

*نویسنده مسئول: a.karimi@ihu.ac.ir

دو کلاس مستعد خطا و بدون مستعد خطا طبقه‌بندی می‌کند و به آن مدل طبقه‌بندی نیز گفته می‌شود. مدل‌هایی که به طبقه‌بندی ماژول‌ها می‌پردازند، ماژول‌ها را بر اساس مستعد خطا بودن آن‌ها مرتب می‌کنند [۶]. این روش می‌تواند با تخصیص بهینه منابع پروژه به ماژول‌های مستعد خطا، به‌صورت مؤثرتری به فرآیند بهبود کیفیت نرم‌افزار که درواقع هدف پیش‌بینی خطای نرم‌افزار است، کمک کند [۷].

کیفیت محصولات نرم‌افزاری به‌شدت با عدم وجود نقص‌ها در ارتباط است. به‌این‌ترتیب برای سازندگان و مدیران پروژه‌های نرم‌افزاری، اطمینان از ساخت نرم‌افزارهایی با کمترین خطای ممکن، بسیار حائز اهمیت است. برای افزایش دقت مدل‌های پیش‌بینی خطای نرم‌افزار، باید ویژگی‌های مؤثر و مرتبط از مجموعه داده‌های در دسترس انتخاب گردند. برای این منظور روش‌های متعددی مطرح‌شده است که می‌توان به روش‌های فیلتر[1] و پوشش[2] اشاره نمود. روش‌های فیلتر روش‌های آماری هستند که جزء روش‌های بدون ناظر محسوب می‌شوند. مهم‌ترین ضعف این روش‌ها فرض مستقل بودن ویژگی‌ها است. به عبارتی، در این روش‌ها اگر وابستگی بین داده‌ها وجود داشته باشد، کارایی روش پایین می‌آید. ازاین‌رو، روش‌های پوشش ابداع شدند. روش‌های پوشش جزء روش‌های با ناظر محسوب می‌شوند که بر اساس قوانین طبیعت به دنبال یافتن زیرمجموعه‌ای مناسب از ویژگی‌ها هستند [۸].

تشخیص خطاهای نرم‌افزار یک فرآیند پیچیده است و ممکن است با چالش‌های مختلفی مواجه شود [۹]. در ادامه تعدادی از این چالش‌ها را شرح می‌دهیم:

پیدا کردن خطاهای مخفی: بعضی از خطاها ممکن است در شرایط خاصی رخ دهند و به‌طورمعمول در زمان آزمون و بررسی‌های معمول قابل‌تشخیص نباشند. این خطاها را خطاهای مخفی یا خطاهای محیطی می‌نامند و تشخیص آن‌ها ممکن است زمان‌بر و دشوار باشد[۹].

تشخیص خطاهای پیچیده: بعضی از خطاها ممکن است باتوجه‌به پیچیدگی و حجم بزرگ نرم‌افزار، به‌سختی قابل‌تشخیص باشند. این خطاها ممکن است به‌صورت تدریجی و در زمان‌های مختلف رخ دهند که تشخیص آن‌ها نیازمند استفاده از روش‌های پیشرفته آزمون و خطایابی است [۹].

تشخیص خطاهای ترکیبی: هنگامی‌که نرم‌افزار با دستگاه‌های دیگری مثل سیستم‌عامل، پایگاه داده، یا نرم‌افزارهای دیگر ادغام می‌شود، خطاهای احتمالی در این فرآیند رخ می‌دهند. تشخیص و رفع این خطاها نیازمند توجه به جزئیات فنی و تعاملات بین اجزای مختلف نرم‌افزار است [۹].

تشخیص خطاهای عملکردی: خطاهای عملکردی مربوط به عدم عملکرد صحیح نرم‌افزار در شرایط مختلف است. این خطاها ممکن است به دلیل نقص در الگوریتم‌ها، برنامه‌نویسی نادرست یا خطاهای پایگاه داده رخ دهند. تشخیص این خطاها نیازمند آزمون‌های جامع و سامانمند نرم‌افزار است [۹].

تشخیص خطاهای ناشناخته: در بعضی موارد، خطاهای ناشناخته و غیرمنتظره ممکن است رخ دهند که تشخیص آن‌ها دشوار است. این خطاها ممکن است به دلیل تداخل‌های پیچیده بین اجزای نرم‌افزار، تغییرات در محیط اجرایی، یا عدم پیش‌بینی مسائل غیرمنتظره باشند [۹].

تشخیص خطاهای پراکنده: بعضی از خطاها ممکن است در بخش‌های مختلف نرم‌افزار پراکنده شوند و در زمان آزمون و بررسی‌های معمول به‌سادگی قابل‌تشخیص نباشند. تشخیص این خطاها نیازمند استفاده از روش‌های پویا و آزمون‌های گسترده است [۹].

برای تشخیص خطاهای نرم‌افزار، استفاده از روش‌های آزمون و بررسی مناسب، استفاده از ابزارهای خودکار آزمون و تجربه و تخصص آزمونگرها و توسعه‌دهندگان بسیار مهم است[۱۰].

در این مقاله، جهت پیش‌بینی خطاهای نرم‌افزار از ترکیب الگوریتم اسب وحشی و نظریه آشوب بهره گرفته خواهد شد. به عبارتی، از نظریه آشوب جهت تولید جمعیت اولیه در اسب وحشی استفاده می‌شود. همچنین در این مقاله از طبقه‌بندی درخت تصمیم جهت طبقه‌بندی خطای نرم‌افزار استفاده خواهد گردید.

در ادامه و در بخش دوم، مروری بر ادبیات تحقیق ارائه می‌شود، سپس در بخش سوم، پیشینه تحقیق را موردبررسی قرار می‌دهیم، در بخش چهارم نیز به ارائه روش پیشنهادی می‌پردازیم. در بخش پنجم ارزیابی روش پیشنهادی تشریح می‌شود و درنهایت، در بخش ششم نتیجه‌گیری و کارهای آینده مرتبط با این تحقیق بیان می‌گردد.

۲. مرور ادبیات تحقیق

در این قسمت لازم است با برخی از مفاهیم اولیه مرتبط با تحقیق جاری آشنا شویم. این مفاهیم شامل خطای نرم‌افزار، عملیات انتخاب ویژگی، نظریه آشوب لجستیک، طبقه‌بندی درخت تصمیم و عملیات طبقه‌بندی است. در ادامه این مفاهیم موردبحث و بررسی قرار خواهند گرفت.

[1] Filter.
[2] Wrapper.

۲-۱. خطای نرم‌افزار

امروزه باتوجه‌به رشد نرم‌افزار ازنظر اندازه و پیچیدگی، حفظ کیفیت نرم‌افزار از اهمیت ویژه‌های برخوردار است [۱۱]. آزمون نرم‌افزار، یکی از مراحل مهم در توسعه نرم‌افزار است که باهدف شناسایی خطاهای برنامه، انجام می‌شود. خطای نرم‌افزار به هر نوع خلل، نقص یا خطا در عملکرد یک نرم‌افزار اشاره دارد. این خطاها می‌توانند به اشکال مختلفی رخ دهند و می‌توانند توسط کاربران، توسعه‌دهندگان یا آزمونگرها شناسایی شوند. خطاهای نرم‌افزار می‌توانند ناشی از مشکلات در طراحی، برنامه‌نویسی، آزمون یا اجرای نرم‌افزار باشند [۱۲].

به‌طورکلی، خطاهای نرم‌افزار را می‌توان به دودسته خطاهای سخت‌افزاری و خطاهای نرم‌افزاری تقسیم کرد [۱۳]. مشکلات سخت‌افزاری عمدتاً به مشکلات فیزیکی مرتبط با سخت‌افزار مانند خرابی قطعات یا عدم تطابق سخت‌افزار با نرم‌افزار اشاره دارند. از سوی دیگر، خطاهای نرم‌افزاری بیشتر به مشکلات مرتبط با کدهای برنامه، عدم صحت الگوریتم‌ها، خطاهای منطقی یا نواقص در طراحی نرم‌افزار مرتبط هستند [۱۴]. خطاهای نرم‌افزار می‌توانند در انواع مختلفی باشند [۱۵]، ازجمله:

- **خطاهای عملکردی:** این خطاها مربوط به عدم عملکرد صحیح نرم‌افزار در شرایط مختلف هستند. به‌عنوان‌مثال، نرم‌افزار ممکن است به درخواست‌های کاربران پاسخ ندهد یا نتایج نادرست را نمایش دهد.
- **خطاهای منطقی:** این خطاها معمولاً به دلیل نقص در منطق و الگوریتم‌های استفاده‌شده در نرم‌افزار رخ می‌دهند. به‌عنوان‌مثال، خطاهای منطقی ممکن است باعث تولید نتایج ناصحیح یا خطاهای محاسباتی شوند.
- **خطاهای وابسته به سخت‌افزار:** این خطاها ممکن است به دلیل تداخل با سخت‌افزار یا تنظیمات نادرست سخت‌افزار رخ دهند. به‌عنوان‌مثال، نرم‌افزار ممکن است با سیستم‌عامل، گرداننده‌ها[1] یا دستگاه‌های سخت‌افزاری ناسازگاری داشته باشد.
- **خطاهای امنیتی:** این خطاها مربوط به ضعف‌ها و آسیب‌پذیری‌های امنیتی در نرم‌افزار هستند. ممکن است باعث نفوذ و دسترسی غیرمجاز به سیستم شوند.
- خطاهای کارایی: این خطاها ممکن است به دلیل کارایی نادرست یا ناکارآمدی نرم‌افزار در مصرف منابع سیستمی (مانند حافظه، پردازنده یا شبکه) رخ دهند.
- **خطاهای واسط کاربری:** این خطاها ممکن است به دلیل طراحی ناصحیح واسط کاربری رخ دهند که باعث مشکلات در تعامل کاربر با نرم‌افزار می‌شود.

[1] Drivers.

تشخیص و رفع خطاهای نرم‌افزار از طریق فرآیندهای آزمون و خطایابی ممکن است انجام شود [۱۶]. استفاده از فنون و ابزارهای مناسب، تجربه آزمونگرها و توسعه‌دهندگان و رعایت استانداردها و بهترین شیوه‌های برنامه‌نویسی نیز می‌تواند در کاهش خطاهای نرم‌افزار مؤثر باشد [۱۷]. پیش‌بینی خطا، یک حوزه تحقیقاتی مؤثر در زمینه مهندسی نرم‌افزار است. فنون و معیارهای بسیاری برای بهبود عملکرد پیش‌بینی خطاها توسعه‌یافته‌اند. در دهه‌های اخیر، مطالعات و تحقیقات متعددی، حوزه پیش‌بینی خطای نرم‌افزاری را موردبررسی قرار داده‌اند [۱۴].

۲-۲. انتخاب ویژگی

انتخاب ویژگی در حوزه پیش‌بینی خطاهای نرم‌افزار بسیار مهم است. انتخاب ویژگی‌های مفید و مؤثر، می‌تواند تأثیر مستقیمی بر کارایی و دقت مدل پیش‌بینی خطا داشته باشد [۱۸]. در ادامه، چند نکته برای انتخاب ویژگی مناسب در پیش‌بینی خطاهای نرم‌افزار ذکر می‌شود:

- **انتخاب ویژگی‌های مرتبط با مسئله:** ویژگی‌های منتخب باید با مسئله پیش‌بینی خطاهای نرم‌افزار مرتبط باشند. به‌عنوان‌مثال، ویژگی‌هایی که نشان‌دهنده الگوها یا خصوصیات خاص خطاها هستند، می‌توانند برای پیش‌بینی خطاها مفید باشند.
- **انتخاب ویژگی‌های معنادار:** ویژگی‌های منتخب باید اطلاعات مفیدی را درباره خطاهای نرم‌افزار ارائه دهند. ویژگی‌های بی‌معنی یا تکراری می‌توانند منجر به افزایش پیچیدگی و کاهش دقت مدل پیش‌بینی خطا شوند.
- **انتخاب ویژگی‌های قابل‌استفاده و قابل‌دسترسی:** ویژگی‌هایی که قابلیت استفاده واقعی در محیط آزمون و تولید نرم‌افزار رادارند، می‌توانند در پیش‌بینی خطاها مؤثرتر باشند. همچنین، ویژگی‌هایی که به‌راحتی و با هزینه کم در دسترس هستند، انتخاب مناسبی هستند.
- **انتخاب ویژگی‌های مستقل از ویژگی‌های دیگر:** ویژگی‌هایی که مستقل از یکدیگر هستند و اطلاعات منحصربه‌فردی را درباره خطاها ارائه می‌دهند، می‌توانند در پیش‌بینی خطاها مؤثرتر باشند. ویژگی‌های تکراری یا وابسته به یکدیگر ممکن است تأثیر مشابهی در پیش‌بینی خطاها داشته باشند و اطلاعات تکراری را ارائه دهند.
- **انتخاب ویژگی‌های قابل تفسیر:** ویژگی‌هایی که قابلیت توضیح و تفسیر دارند و ارتباط معناداری با خطاها دارند، می‌توانند در پیش‌بینی و تحلیل خطاها مفید باشند. ویژگی‌هایی که توصیف‌کننده وضعیت یا خصوصیات خاصی از نرم‌افزار هستند، می‌توانند

اطلاعات قابل‌توجهی را درباره خطاها ارائه دهند[۱۹].

درنهایت، انتخاب ویژگی مناسب برای پیش‌بینی خطاهای نرم‌افزار نیازمند تجربه، دانش و تحلیل دقیق است. این فرآیند ممکن است نیاز به آزمون و ارزیابی چندین زیرمجموعه ویژگی داشته باشد تا بهترین نتیجه را برای پیش‌بینی خطاها به دست آورد [۱۹].

۲-۳. نظریه آشوب

نظریه آشوب شاخه‌ای از علم ریاضیات است که به بررسی رفتار آن دسته از سیستم‌های دینامیکی می‌پردازد که به شرایط اولیه بسیار حساس هستند [۲۰]. آشوب در یک بیان عام به حالت سردرگمی و بی‌نظمی کامل و یا فقدان کامل نظم تعبیر می‌شود، اما یک تضاد جالب در درون مفهوم خود دارد. آشوب درواقع ادعا می‌کند که علمی است برای پیش‌بینی رفتار سیستم‌های ذاتاً غیرقابل‌پیش‌بینی و بنیان اصلی آن این ایده است که نظم و آشوب همیشه مخالف و در مقابل هم نیستند. نظریه آشوب با پخش کردن جمعیت در فضای مسئله، باعث پوشش دهی بهتر مسئله می‌شود. مهم‌ترین نکته در بهره‌گیری از نظریه آشوب، قرار نگرفتن جمعیت‌ها در نزدیکی یکدیگر است. به عبارتی، در این روش سعی می‌گردد تراکم جمعیت‌ها در یک نقطه قرار نگیرد و در تمام فضای مسئله توزیع شود. درروش تصادفی، تراکم جمعیت معمولاً در بخش بالایی فضای مسئله است درصورتی‌که در نظریه آشوب، تراکم جمعیت در تمامی فضای مسئله یکسان است [۲۱]. درروش پیشنهادی، نظریه آشوب باعث می‌گردد جای‌گشت‌های متفاوت و غیرتکراری از ویژگی‌ها انتخاب گردد تا روش پیشنهادی بتواند زیرمجموعه‌ای مناسب از ویژگی‌ها را انتخاب نماید. درصورتی‌که درروش تصادفی، جای‌گشت‌های انتخابی تکراری زیادتر می‌گردد [۲۲]. دلیل دیگر در استفاده از نظریه آشوب در این مقاله این است که نظریه آشوب باعث جلوگیری از قرار نگرفتن الگوریتم اسب وحشی در بهینگی محلی می‌شود. در ادامه و در این مقاله از نظریه آشوب جهت تولید جمعیت اولیه در الگوریتم اسب وحشی استفاده‌شده است.

۲-۴. طبقه‌بندی

در دنیای امروز موضوع طبقه‌بندی اطلاعات اهمیت بسیاری دارد. این‌که بتوان مدلی مناسب برای تحلیل داده‌هایی خاص به دست آورد و با بررسی اولیه ویژگی‌های یک عنصر خاص، الگوی رفتاری آن عنصر را پیش‌بینی کرد بسیار حائز اهمیت است [۲۳].

در طبقه‌بندی اطلاعات، هدف به دست آوردن مدلی برای الگوی رفتاری و ویژگی‌هایی از داده‌ها است تا با کمک آن بتوان بدون دانستن رفتار یک موجودیت، باتوجه‌به ویژگی‌های آن و با استفاده از مدل به‌دست‌آمده، رفتار آن را تشخیص داد و آن موجودیت را در گروه خاصی طبقه‌بندی کرد. فرآیند طبقه‌بندی، یک فرآیند دومرحله‌ای است که در مرحله نخست با کمک مجموعه داده‌های آموزشی[1] که برچسب کلاس تمام نمونه‌های آن مشخص است، مدل ساخته می‌شود. این مرحله به نام مرحله یادگیری شناخته می‌شود. در مرحله دوم با کمک مجموعه داده‌های آزمون[2] که در آن معمولاً برچسب کلاس‌ها نامعلوم است، مدل به‌دست‌آمده اعتبارسنجی می‌شود. درواقع ارزش‌یابی مدل مرتبط است به اینکه کلاس، چه تعداد از نمونه داده‌های آزمون را به‌درستی تخمین زده است. ازجمله فنونی که برای طبقه‌بندی به کار می‌رود می‌توان به درخت‌های تصمیم‌گیری، ماشین بردار پشتیبان و نزدیک‌ترین همسایه و غیره اشاره کرد [۲۴].

۲-۴-۱. درخت تصمیم‌گیری

طبقه‌بندی درخت تصمیم یکی از شناخته‌شده‌ترین فنون یادگیری ماشین است [۲۵]. درخت تصمیم از گره‌های تصمیم و گره‌های برگ ساخته‌شده است. هر گره تصمیم متناظر با یک آزمون X بر روی یک مشخصه از داده‌های ورودی است و تعدادی شاخه دارد که هرکدام نتیجه آزمون X را کنترل می‌کنند. هر گره برگ نشان‌دهنده یک کلاس است که نتیجه تصمیم‌گیری برای یک مورد است [۲۶].

هنگامی‌که یک درخت تصمیم ساخته شد، می‌توان از آن برای طبقه‌بندی داده‌های آزمون استفاده کرد که دارای ویژگی‌های مشابه داده‌های آموزشی هستند. در درخت تصمیم با شروع از گره ریشه درخت، آزمون بر روی همان ویژگی انجام می‌شود که گره ریشه نشان می‌دهد. فرآیند تصمیم‌گیری، شاخه‌ای را در برمی‌گیرد که شرایط آن توسط مقدار ویژگی آزمون شده برآورده می‌شود. این شاخه، فرآیند تصمیم‌گیری را به فرزند گره ریشه هدایت می‌کند. همین فرآیند به‌صورت بازگشتی اجرا می‌شود تا زمانی که به یک گره برگ برسیم. [۲۷].

۳. پیشینه تحقیق

در این بخش، به تشریح تحقیقات انجام‌گرفته توسط سایر پژوهشگران در زمینهٔ پیش‌بینی خطای نرم‌افزار می‌پردازیم. هر یک از این کارها در بخش‌های جداگانه به‌صورت زیر تشریح می‌شود. در ادامه، خلاصه‌ای از هدف، مزایا و معایب هر یک از کارهای انجام‌شده، در جدول (۱) آورده شده است.

[1] Testing Dataset.
[2] Traning Dataset.

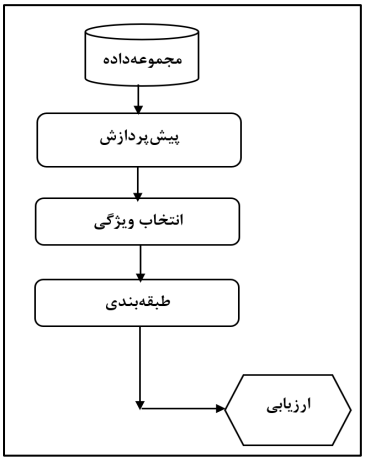
جدول (۱). مقایسه کارهای انجام‌شده

ارائه‌دهنده راه‌حل	هدف	مزایا	معایب
تومار و همکاران [۲۸]	رفع مشکلات طبقه‌بندی و ایجاد یک سیستم دقیق برای پیش‌بینی خطاهای نرم‌افزار	LDA از سایر طبقه‌بندی‌ها بهتر عمل می‌کند. طبقه‌بندی KNN بهترین زمان اجرا را دارد.	- عدم درنظرگرفتن اهمیت ویژگی‌ها برای افزایش عملکرد طبقه‌بندی‌ها - به دست نیاوردن دقت مدل
طاهر و همکاران [۲۹]	کاهش ابعاد مجموعه داده با استفاده از فن انتخاب ویژگی	روش پیشنهادی نسبت به الگوریتم بهینه‌سازی شاهین هریس اصلی برتر است.	عدم بررسی کارایی سایر فنون دوبعدی و همچنین نسخه‌های دیگر TFS
حسونه و همکارانش [۳۰]	رسیدگی به مشکلات مرتبط با انتخاب ویژگی	- ارتقاء کیفیت الگوریتم وال برتر - اجتناب از بهینگی محلی	زمان محاسباتی SVM بالاترین بود.
الشغایر و همکاران [۳۱]	پیش‌بینی خطای نرم‌افزار	بهبود عملکرد سیستم پیش‌بینی خطای نرم‌افزار	برای مجموعه داده مقیاس کوچک کافی نیست.
احمد و همکاران [۳۲]	پیش‌بینی ماژول‌های دارای نقص نرم‌افزار با استفاده از سه مجموعه داده NASA	بهبود عملکرد سیستم پیش‌بینی خطای نرم‌افزار در زمان واقعی از برنامه‌های کاربردی موردنظر	- عدم استفاده از طبقه‌بندی‌های گروهی یا ترکیبی KNN- پایین‌ترین مقدار را بر روی مجموعه داده‌های PC1 انجام داد.
قاسم و همکاران [۳۳]	بررسی عوامل مؤثر بر عملکرد الگوریتم‌های یادگیری عمیق موردمطالعه در حوزه یادشده.	تأثیر پارامترهای اصلاح‌کننده نقش مهمی در عملکرد پیش‌بینی دارد	عدم استفاده از مجموعه داده‌های دیگر

همان‌طور که اشاره شد، در این بخش برخی از تحقیقات انجام‌شده معتبر در زمینهٔ پیش‌بینی خطای نرم‌افزار موردبررسی قرار گرفت. یکی از اهداف اصلی در بررسی کارهای مرتبط، اخذ ایده‌های مختلف در زمینه افزایش دقت پیش‌بینی خطای نرم‌افزار است. جهت افزایش دقت پیش‌بینی، از روش‌های متعددی می‌توان استفاده نمود که از فنون مختلفی نظیر انتخاب ویژگی با ترکیب الگوریتم اسب وحشی و نظریه آشوب، طبقه‌بندی و غیره می‌توان بهره برد.

۴. روش پیشنهادی

در این مقاله، جهت پیش‌بینی خطای نرم‌افزار از یک سازوکار چندمرحله‌ای بهره گرفته‌شده است که مراحل آن در ادامه بیان‌شده است.



شکل (۱). فرآیند پیش‌بینی خطای نرم‌افزار

همان‌طور که در شکل (۱) نشان داده‌شده است، جهت پیش‌بینی خطای نرم‌افزار از یک سازوکار چندمرحله‌ای بهره گرفته‌شده است. در مرحله نخست، عملیات پیش‌پردازش به روی‌داده‌ها صورت خواهد گرفت. هدف از مرحله پیش‌پردازش افزایش کیفیت داده‌ها است. در این بخش عملیاتی نظیر اصلاح داده گمشده، نرمال‌سازی و تقسیم‌بندی داده صورت می‌گیرد. در گام دوم عملیات انتخاب ویژگی انجام می‌گیرد. هدف از عملیات انتخاب ویژگی، انتخاب زیرمجموعه‌ای از ویژگی‌های مؤثر و بهینه است که بتواند دقت سیستم پیشنهادی در پیش‌بینی خطای نرم‌افزار را افزایش دهد. در این مقاله، جهت عملیات انتخاب ویژگی از روش‌های پوشش استفاده گردیده است. الگوریتم مورداستفاده در این مقاله، الگوریتم اسب وحشی است. مهم‌ترین ضعف الگوریتم‌های تکاملی، قرار گرفتن آن‌ها در بهینگی محلی است. برای رفع مشکل فوق، از نظریه آشوب استفاده‌شده است. درنهایت، جهت طبقه‌بندی و پیش‌بینی خطای نرم‌افزار از روش‌های یادگیری ماشین استفاده‌شده است. در ادامه هر یک از بخش-های موردنظر شرح داده خواهد شد.

۴-۱. پیش‌پردازش داده‌ها

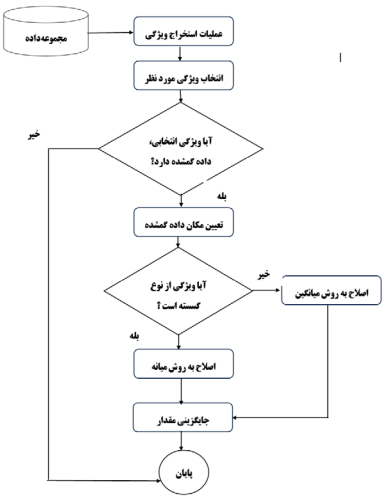
هدف از عملیات پیش‌پردازش داده‌ها، بهبود داده‌ها است. جهت بهبود داده‌ها در این مقاله از سه روش اصلاح داده گمشده، نرمال‌سازی و تقسیم‌بندی داده استفاده‌شده است. در ادامه، هر یک از زیر بخش‌های موردنظر شرح داده خواهد شد.

۴-۱-۱. اصلاح داده گمشده

هدف از اصلاح داده گمشده تنظیم مقادیری است که اطلاعات آن‌ها در دسترس نباشد. دلایل وجود داده‌های گمشده در مجموعه داده به شرح زیر است.

- **برخورداری از مقدار نامعتبر**: ممکن است مقادیری در مجموعه داده وجود داشته باشند که به‌صورت نامعتبر یا خالی واردشده‌اند.
- **خطا در فرآیند جمع‌آوری داده**: هنگام جمع‌آوری داده، ممکن است به دلایلی مانند اشتباه در تایپ، انتقال داده نادرست و غیره، مقادیری از دست بروند.
- **حذف داده‌ها**: در برخی موارد، به‌منظور حفظ حریم شخصی، داده‌هایی از مجموعه حذف می‌شوند. این عمل باعث ایجاد داده گمشده می‌گردد.
- **خرابی سیستم**: در صورت بروز خرابی در فرآیند جمع‌آوری و ذخیره داده، ممکن است برخی از داده‌ها به‌صورت نامعتبر در مجموعه داده قرار گیرند.
- **مشکلات در استخراج داده**: ممکن است در فرآیند استخراج داده از منابع مختلفی مانند پایگاه داده‌ها، وب‌سایت‌ها و غیره، مقادیری از داده‌ها از دست بروند.

اصلاح داده گمشده در فرآیند یادگیری باعث بهبود دقت و کارایی الگوریتم، حذف نیاز به جستجوی داده‌های جایگزین، افزایش تعداد نمونه‌ها و بهبود فرآیند تصمیم‌گیری می‌گردد. ازاین‌رو، در این مقاله از روش‌های نشان داده‌شده در شکل (۲) جهت بهبود داده‌های گمشده استفاده می‌گردد.



شکل (۲). فرآیند اصلاح داده گمشده

در شکل (۲)، فرآیند اصلاح داده گمشده نشان داده‌شده است. در این فرآیند، عملیات اصلاح داده گمشده بر اساس روش‌های میانگین و میانه صورت گرفته است. اگر مقدار ویژگی از نوع گسسته باشد، جهت اصلاح داده گمشده از روش میانه استفاده می‌گردد. در این حالت مقادیر ویژگی که دارای مقادیر گمشده نیستند در یک بردار قرار می‌گیرند. سپس، مقادیر بر اساس حالت صعودی مرتب می‌شوند. در این حالت از میانه‌ی بردار برای جایگزینی مقدار موردنظر استفاده می‌گردد. در رابطه (۱) این عملیات نشان داده‌شده است.

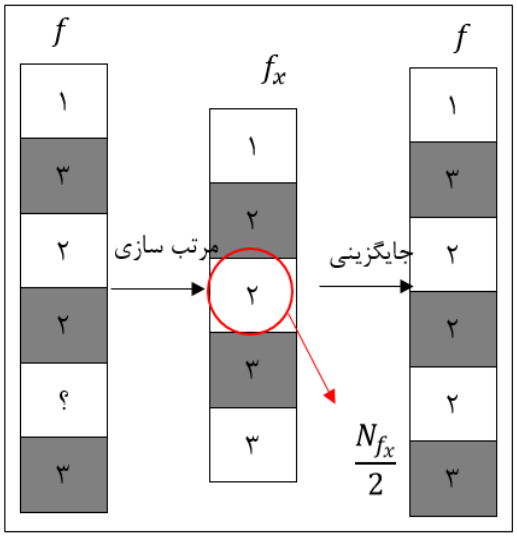
$$\text{Middle}(f) = \begin{cases} f_i & if\ N_{f_x}\ is\ odd \quad i = \frac{N_{fx}}{2} \\ \frac{f_i + f_{i+1}}{2} & if\ N_{f_x}\ is\ even \end{cases} \quad (۱)$$

در رابطه (۱)، $Middle(f)$ بیانگر میانه‌ی ویژگی fام، N_{f_x} بیانگر تعداد مقادیری از ویژگی f که دارای مقدار گمشده نیست، f_i بیانگر iامین مقدار از مجموعه داده غیر گمشده است که مقدار آن برابر $\frac{N_{fx}}{2}$ است.

اگر مقدار ویژگی از نوع پیوسته باشد، برای اصلاح داده گمشده از روش میانگین استفاده می‌گردد. در رابطه (۲) این عملیات نشان داده‌شده است.

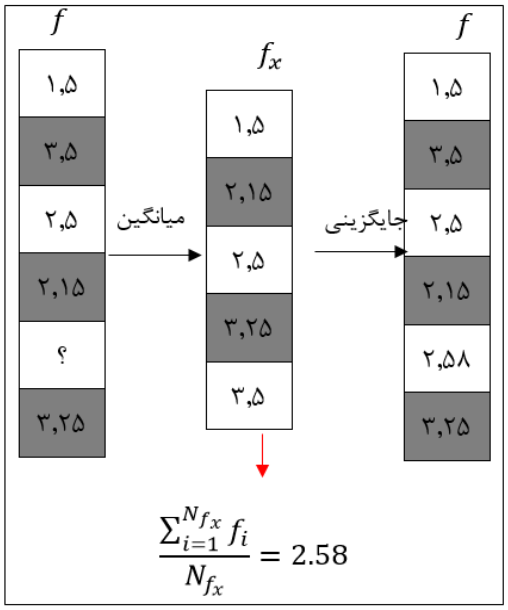
$$\text{Median}(f) = \frac{\sum_{i=1}^{N_{fx}} f_i}{N_{fx}} \quad (۲)$$

در رابطه (۲)، $Median(f)$ بیانگر مقدار میانگین ویژگی fاست. در شکل (۳)، عملیات اصلاح داده گمشده به روش میانه و میانگین نشان داده‌شده است.



شکل (۳). اصلاح داده گمشده در حالت گسسته

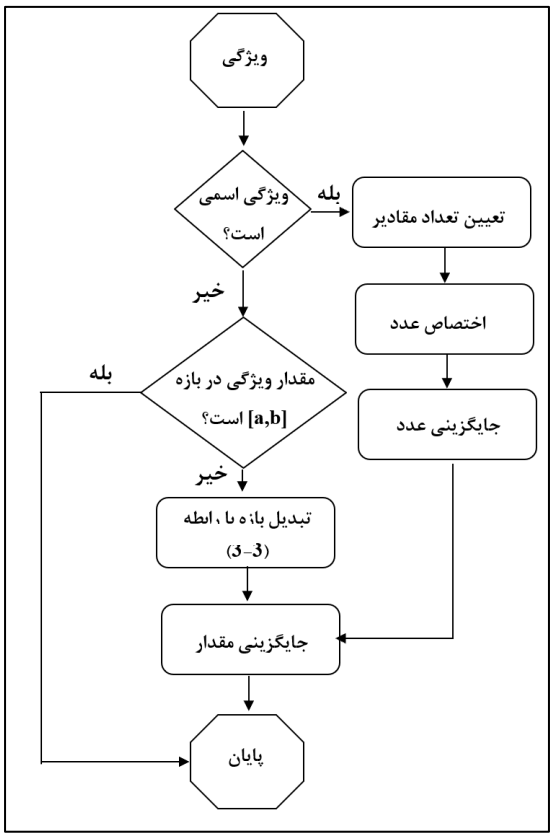
اگر مقدار ویژگی پیوسته باشد، در این حالت برای اصلاح داده گمشده از روش میانگین استفاده می‌شود. عملیات اصلاح داده گمشده به روش میانگین در شکل (۴) نشان داده‌شده است.



شکل (۴). اصلاح داده گمشده در حالت پیوسته

۴-۱-۲. نرمال‌سازی

هدف از عملیات نرمال‌سازی، تغییر مقدار یک ویژگی از یک بُعد به بُعد دیگر است. در شکل (۵) فرآیند نرمال‌سازی نشان داده‌شده است.



شکل (۵). فرآیند نرمال‌سازی داده

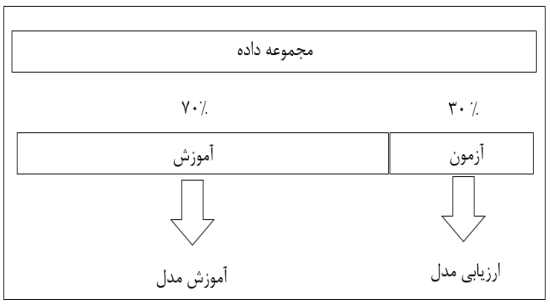
در شکل (۵)، فرآیند نرمال‌سازی داده نشان داده‌شده است. در این فرآیند از دو نرمال‌سازی اسمی و مقداری استفاده‌شده است. در حالت نرمال‌سازی اسمی، به هر مقدار اسمی یک عدد منحصربه‌فرد اختصاص داده می‌شود. بعد از عملیات اختصاص، عددهای موردنظر بجای مقادیر اسمی در مجموعه داده قرار می‌گیرند. در نرمال‌سازی مقدار، جهت تبدیل بازه‌ی ویژگی از بُعد *[a,b]* به بُعد *[c,d]* از رابطه (۳) استفاده می‌گردد.

$$\text{Norm(f)} = \frac{f_x - a}{b-a} * (d-c) + c \qquad (۳)$$

در رابطه (۳)، $Norm(f)$ بیانگر مقدار نرمال‌شده ویژگی f، f_x بیانگر مقدار ویژگی، a بیانگر حداقل مقدار ویژگی قبل از عملیات نرمال‌سازی، b بیانگر حداکثر مقدار ویژگی قبل از عملیات نرمال‌سازی، c بیانگر مقدار ویژگی بعد از عملیات نرمال‌سازی و d بیانگر مقدار ویژگی بعد از عملیات نرمال‌سازی است.

۴-۱-۳. تقسیم‌بندی داده

هدف از این بخش، تقسیم‌بندی داده‌ها به داده‌های آموزش و آزمون است. از داده‌های آموزش برای استخراج ویژگی‌های منتخب استفاده می‌شود. از داده‌های آزمون به‌منظور بررسی عملکرد روش انتخاب ویژگی استفاده می‌شود. جهت تقسیم‌بندی داده‌ها به داده‌های آموزش و آزمون از روش‌های تصادفی و چندبخشی استفاده می‌گردد. در حالت تصادفی داده‌ها به دودسته آموزش و آزمون با درصد ۷۰ به ۳۰ تقسیم‌بندی می‌گردد، که ۷۰ درصد داده‌ها مربوط به داده‌های آموزش و ۳۰ درصد داده‌ها مربوط به داده‌های آزمون است. در شکل (۶) این عملیات نشان داده‌شده است.

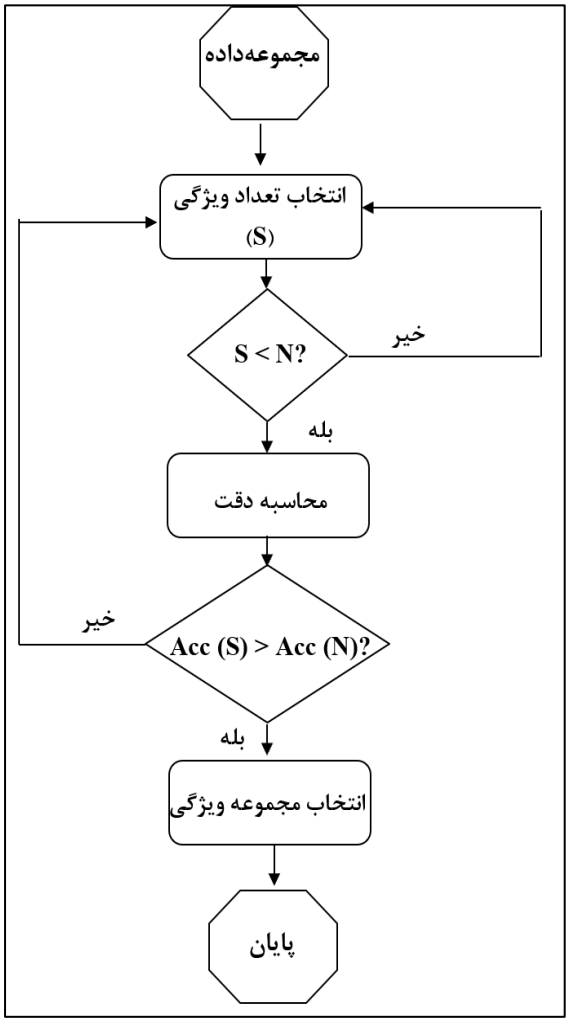


شکل (۶). تقسیم‌بندی داده‌ها به روش تصادفی

در حالت چندبخشی، ابتدا داده‌ها به k بخش تقسیم می‌گردد و در هر تکرار یک بخش به‌عنوان داده آزمون و $k-1$ بخش به‌عنوان داده آموزش در نظر گرفته می‌شود. این فرآیند به تعداد k بار صورت می‌گیرد. درنهایت نتیجه نهایی از میانگین جواب‌های به‌دست‌آمده از هر تکرار به دست می‌آید. در این مقاله از روش تصادفی برای تقسیم‌بندی داده‌ها استفاده‌شده است.

۴-۲. انتخاب ویژگی

یکی از بخش‌های مهم در سیستم‌های پیش‌بینی خطای نرم‌افزار، مرحله انتخاب ویژگی است. زیرا ویژگی‌ها تأثیر مستقیمی بروی کارایی الگوریتم‌های پیش‌بینی دارند. ازاین‌رو، هرچقدر ویژگی-های انتخابی کارآمدتر باشند، کارایی الگوریتم یادگیری و طبقه‌بندی نیز افزایش پیدا می‌کند. فرآیند انتخاب ویژگی در شکل (۷) نشان داده‌شده است.



شکل (۷). فرآیند انتخاب ویژگی

در شکل (۷)، فرآیند انتخاب ویژگی نشان داده‌شده است. مجموعه ویژگی انتخابی باید دو ویژگی نسبت به مجموعه داده اصلی داشته باشد که در ادامه بیان‌شده است:

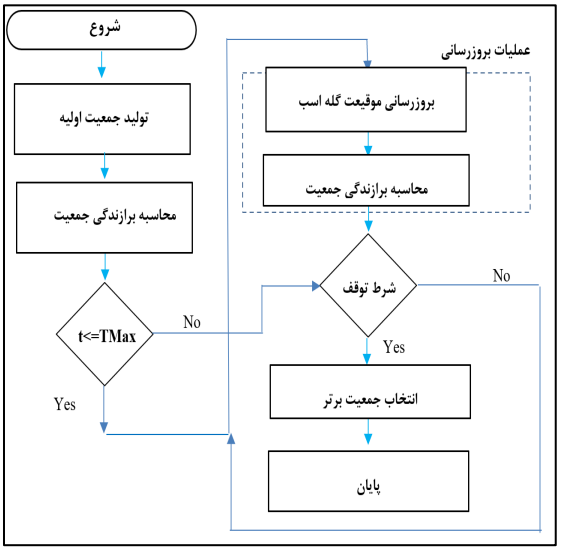
تعداد ویژگی‌ها: در زیرمجموعه ویژگی منتخب، تعداد ویژگی‌ها باید نسبت به مجموعه داده اصلی کمتر باشد. به عبارتی باید شرط $S<N$ برقرار باشد.

دقت: دقت زیرمجموعه ویژگی منتخب باید نسبت به‌دقت مجموعه ویژگی اصلی بیشتر باشد. به عبارتی باید رابطه $Acc(S) > Acc(N)$ برقرار باشد که دراین‌رابطه Acc بیانگر دقت است.

در این مقاله، جهت انجام عملیات انتخاب ویژگی از الگوریتم تکاملی اسب وحشی استفاده‌شده است. در ادامه توضیحات مربوط به این الگوریتم آورده شده است.

۴-۳. انتخاب ویژگی مبتنی بر الگوریتم اسب وحشی

همان‌طور که قبلاً بیان شد در این مقاله، جهت انتخاب ویژگی از الگوریتم تکاملی اسب وحشی استفاده‌شده است. ساختار این الگوریتم در شکل (۸) نشان داده‌شده است.



شکل (۸). فرآیند انتخاب ویژگی توسط الگوریتم اسب وحشی

در شکل (۸)، فرآیند انتخاب ویژگی توسط الگوریتم اسب وحشی نشان داده‌شده است. در این فرآیند از مراحل زیر استفاده‌شده است:

- تولید جمعیت اولیه
- محاسبه تابع برازندگی
- به‌روزرسانی جمعیت
- شرط توقف

در ادامه، توضیحات هر یک از مراحل شرح داده می‌شود.

۴-۳-۱. جمعیت اولیه

جمعیت اولیه نشان‌دهنده راه‌حل‌های مسئله پیشنهادی است که به‌عنوان کاندیدهای راه‌حل مسئله شناخته می‌شود. در الگوریتم-های تکاملی، جمعیت اولیه به‌صورت تصادفی تعیین می‌گردد. رابطه (۴)، تولید جمعیت اولیه به‌صورت تصادفی را بیان می‌کند.

در رابطه (۴)، Pop_i بیانگر جمعیت اولیه، X_r بیانگر مقدار

تصادفی (در بازه‌ی ۰ تا ۱)، Lb بیانگر کران پایین مسئله و UP بیانگر کران بالای مسئله است. در شکل (۹)، شبه کد تولید جمعیت اولیه نشان داده‌شده است.

تابع ایجاد جمعیت اولیه

ورودی: تعداد جمعیت، حد بالا، حد پایین

خروجی: جمعیت اولیه ایجادشده (X)

برای هر جمعیت عملیات زیر را انجام بده

۱-به تعداد ویژگی‌ها اعداد تصادفی در بازه‌ی حد پایین و حد بالا ایجاد کن

۲-مقادیر به‌دست‌آمده را به‌عنوان جمعیت جدید در نظر بگیر

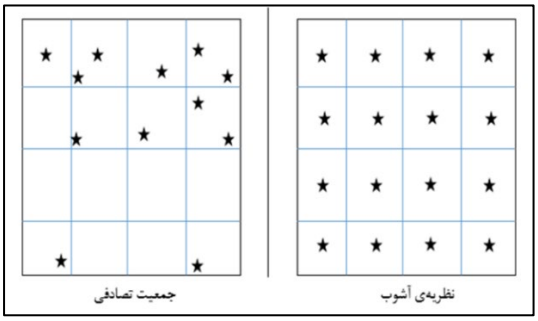
پایان

جمعیت اولیه را بازگردان

شکل (۹). شبه کد عملیات تولید جمعیت اولیه

- **بهبود جمعیت اولیه**

یکی از مهم‌ترین عیب‌های الگوریتم‌های تکاملی در بهینه‌سازی یک مسئله، قرار گرفتن الگوریتم در بهینگی محلی است. زیرا در الگوریتم‌های تکاملی تنوع جواب‌ها حفظ نمی‌گردد. یکی از مهم‌ترین دلایل عدم ایجاد تنوع جواب، تولید تصادفی جمعیت اولیه است. در این مقاله جهت تولید جمعیت اولیه از نظریه آشوب استفاده‌شده است. در شکل (۱۰) تفاوت نظریه آشوب و حالت تصادفی در تولید جمعیت اولیه نشان داده‌شده است.



شکل (۱۰). تولید جمعیت اولیه به روش نظریه آشوب و تصادفی

همان‌طور که در شکل (۱۰) مشاهده می‌شود، در حالت تصادفی تراکم جمعیت در یک نقطه متمرکز است که در این حالت تنوع جواب‌ها حفظ نشده است ولی درروش نظریه آشوب، جمعیت‌ها در سرتاسر فضای مسئله پخش‌شده‌اند، لذا تنوع جواب‌ها در این روش کاملاً مشهود است.

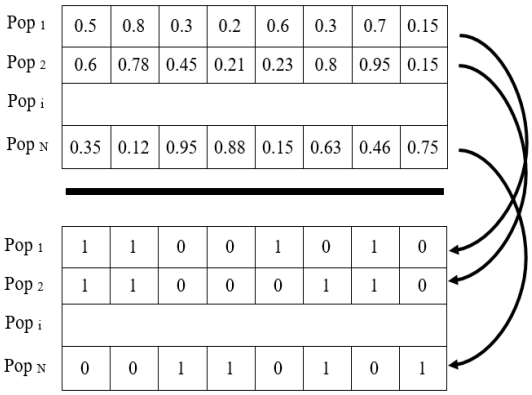
- **دودویی سازی جمعیت اولیه**

مسئله انتخاب ویژگی یک مسئله گسسته است، درحالی‌که الگوریتم تکاملی اسب وحشی یک مسئله پیوسته است؛ بنابراین، جمعیت اولیه باید به حالت دودویی تبدیل گردد. به عبارتی، باید مقادیر پیوسته به مقادیر دودویی تبدیل گردند. در حالت دودویی که فقط مقادیر صفر و یک موجود است، صفر بیانگر عدم انتخاب ویژگی و یک بیانگر انتخاب ویژگی است. در این مقاله، جهت دودویی سازی مقدار جمعیت از رابطه (۵) استفاده‌شده است.

$$T(x_j) = \frac{1}{1+e^{-x_j}} \qquad (۵)$$

$$S_j = \begin{cases} 1 & r < T(x_j) \\ 0 & otherwise \end{cases}$$

در رابطه (۵)، x_j بیانگر مقدار جمعیت در بُعد jام‌است. در شکل (۱۱) عملیات دودویی سازی بر روی جمعیت اولیه صورت گرفته است.



شکل (۱۱). عملیات دودویی سازی بر روی جمعیت اولیه

۴-۳-۲. برازندگی جمعیت

برازندگی جمعیت نشان‌دهنده برتری جمعیت نسبت به جمعیت دیگر است. برازندگی جمعیت به‌عنوان تابع سودمندی نیز شناخته می‌شود. تابع برازندگی می‌تواند در دو حالت به شرح زیر مورداستفاده قرار گیرد:

تک‌هدفه: در این حالت، الگوریتم به دنبال بهینه کردن یک هدف است. هدف موردنظر می‌تواند در دو حالت ماکزیمم و مینیمم مورداستفاده قرار گیرد. در حالت ماکزیمم، الگوریتم به دنبال بیشینه کردن تابع هدف است؛ ولی در حالت مینیمم الگوریتم به دنبال کمینه کردن تابع هدف است.

چندهدفه: در این حالت، الگوریتم به دنبال بهینه کردن چندین هدف است. اهداف می‌توانند موافق با یکدیگر یا متضاد با یکدیگر باشند.

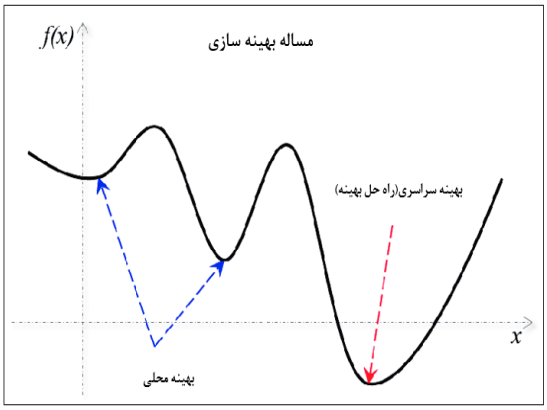
در این مقاله، جهت محاسبه تابع برازندگی از حالت تک‌هدفه استفاده‌شده است که این هدف قرار است کمینه گردد. تابع برازندگی مورداستفاده، در رابطه (۶) آورده شده است.

$$Fitness = \alpha * (1 - ACC) + \beta * (\frac{N_S}{D}) \qquad (۶)$$

در رابطه (۶)، $Fitness$ بیانگر تابع برازندگی، ACC بیانگر دقت روش پیشنهادی، N_S بیانگر تعداد ویژگی‌های انتخابی، D بیانگر تعداد کل ویژگی‌ها و α و β ضریب‌های ترکیب است که مجموع آن‌ها باید برابر یک گردد.

۴-۳-۳. به‌روزرسانی جمعیت

یکی از بخش‌های مهم هر الگوریتم تکاملی، مسئله به‌روزرسانی جمعیت اولیه است. به‌روزرسانی جمعیت، نقش کلیدی در قرار نگرفتن الگوریتم در بهینگی محلی دارد. در شکل (۱۲) شرایط بهینگی محلی نشان داده‌شده است.



شکل (۱۲). شرایط بهینگی محلی

برای قرار نگرفتن الگوریتم‌های تکاملی در بهینگی محلی از دو روش، به شرح زیر استفاده می‌گردد:

عملیات اکتشاف: هدف از این عملیات جستجوی تصادفی فضای مسئله است. در الگوریتم تکاملی اسب وحشی، عملیات اکتشاف درحرکت اسب‌ها به سمت نماینده گله خود است. به عبارتی، در این حالت اسب‌ها به سمت نماینده گله خود حرکت می‌کنند که این حرکت به‌منظور جستجوی فضای بهینه است.

عملیات بهره‌برداری: هدف از این عملیات حرکت به سمت جواب بهینه است. در الگوریتم تکاملی اسب وحشی نماینده هر گروه به سمت بهترین نماینده گروه‌ها حرکت می‌کند. به عبارتی، در این حالت نماینده قصد انتخاب موقعیت بهینه رادارند.

۴-۳-۴. شرط توقف

هدف از این بخش، خاتمه روند بهینه‌سازی است. در این مقاله جهت خاتمه روند بهینه‌سازی، از سه روش زیر استفاده‌شده است.

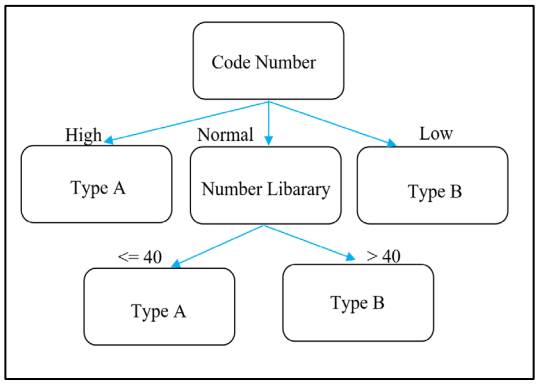
رسیدن به جواب بهینه: در این حالت، الگوریتم زمانی خاتمه می‌یابد که به جواب موردنظر رسیده باشد. برای مثال، اگر خطای انتخاب ویژگی به ۰,۲ درصد برسد، در این حالت الگوریتم روند بهینه‌سازی را متوقف می‌سازد.

عدم بهبود در k تکرار: در این حالت، الگوریتم اگر در k تکرار نتواند مقادیر خود را بهینه کند، الگوریتم روند بهینه‌سازی را خاتمه می‌دهد. ازآنجایی‌که الگوریتم‌های تکاملی در بهینگی محلی قرار می‌گیرند، بنابراین این روش نمی‌تواند مفید واقع شود.

ماکزیمم تکرار: در این حالت، الگوریتم زمانی خاتمه می‌یابد که به تکرار ماکزیمم رسیده باشد. در این مقاله از روش ماکزیمم تکرار استفاده‌شده است.

۴-۴. طبقه‌بندی

جهت پیش‌بینی خطای نرم‌افزار از الگوریتم درخت تصمیم استفاده‌شده است. مشخصات این طبقه‌بندی در شکل (۱۳) آورده شده است.



شکل (۱۳). پیش‌بینی خطای نرم‌افزار توسط الگوریتم درخت تصمیم

مزایای استفاده از درخت تصمیم جهت پیش‌بینی خطای نرم‌افزار در ادامه آورده شده است.

سادگی و قابل‌فهم بودن: درخت تصمیم به دلیل ساختار ساده‌ای که دارد، برای بسیاری از افراد قابل‌فهم و قابل‌استفاده است.

قابلیت تغییر: درخت تصمیم به‌راحتی به‌روزرسانی و تغییر داده می‌شود، بنابراین می‌تواند با تغییرات در محیط نرم‌افزار و مقدار داده‌های ورودی سازگار باشد.

قابلیت استفاده در زمان واقعی: شبکه‌های عصبی و الگوریتم‌های دیگر برای پیش‌بینی خطا، معمولاً زمان بیشتری نسبت به درخت تصمیم به طول می‌انجامند و مناسب برای استفاده در زمان واقعی نیستند. درخت تصمیم به دلیل سادگی، قابلیت استفاده در زمان واقعی را دارد.

قابلیت پیش‌بینی خطاهای چندگانه: درخت تصمیم قابلیت پیش‌بینی خطاهای چندگانه با استفاده از چند شاخه و تصمیمات مختلف را دارا است.

کاهش هزینه و زمان برای پیش‌بینی خطا: با استفاده از درخت تصمیم، هزینه و زمان موردنیاز برای پیش‌بینی خطا کاهش می‌یابد

و می‌توان به‌صورت مؤثرتری خطاها را پیش‌بینی کرد.

۵. ارزیابی روش پیشنهادی

در این بخش، ابتدا مشخصات محیط شبیه‌سازی از منظر سخت‌افزاری و نرم‌افزاری بیان می‌گردد. بعد از معرفی مشخصات محیط شبیه‌سازی، معیارهای ارزیابی آورده شده است. معیارهای ارزیابی در این مقاله شامل معیارهای دقت، صحت، فراخوان و معیار F است. همچنین، مشخصات مجموعه داده به‌منظور آموزش و آزمون مدل طبقه‌بندی آورده شده است. درنهایت، نتایج ارزیابی روش پیشنهادی با دیگر روش‌ها، به‌ویژه مقایسه با مقاله مرجع هم موردبررسی قرارگرفته است. هدف از این ارزیابی، نشان دادن کارایی وبرتری روش پیشنهادی در پیش‌بینی ماژول‌های مستعد خطا است. در ادامه، جزئیات موارد بیان‌شده در بالا به‌صورت کامل آورده شده است.

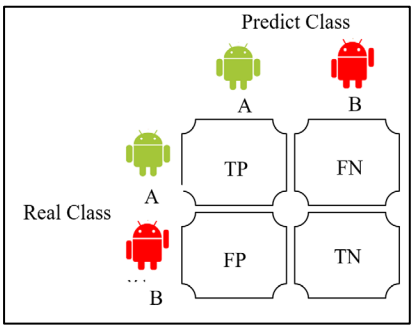
۵-۱. محیط شبیه‌سازی

مشخصات محیط شبیه‌سازی مورداستفاده در این مقاله، از منظر سخت‌افزاری و نرم‌افزاری به شرح زیر است.

۱. نرم‌افزار محیط مدل‌سازی، شامل نرم‌افزار متلب نسخه *2016a* است.
۲. سیستم‌عامل مورداستفاده شامل ویندوز ۱۰، *64 bit* است.
۳. مشخصات سخت‌افزار مورداستفاده شامل *CPU: Core i7 8500* و *RAM: 16 GB* است.

۵-۲. معیارهای ارزیابی

در این بخش، معیارهای ارزیابی روش پیشنهادی که شامل معیارهای دقت، صحت، فراخوان و معیار F است، موردبررسی قرار می‌گیرد. برای محاسبه معیارهای ارزیابی از ماتریس کانفیوژن بهره گرفته‌شده است. مشخصات این ماتریس در شکل (۱۴) نشان داده‌شده است.



شکل (۱۴). ماتریس کانفیوژن

در ماتریس کانفیوژن عبارات *TP*، *FP*، *FN* و *TN* وجود دارد که توضیح این عبارت در ادامه آورده شده است.

TP: تعداد نمونه‌هایی است که کلاس واقعی آن‌ها از نوع A بوده و روش پیشنهادی نیز به‌درستی کلاس A تشخیص داده است.

FP: تعداد نمونه‌هایی است که کلاس واقعی آن‌ها از نوع B بوده؛ ولی روش پیشنهادی به‌اشتباه کلاس آن را A تشخیص داده است.

FN: تعداد نمونه‌هایی است که کلاس واقعی آن‌ها از نوع A بوده؛ ولی روش پیشنهادی به‌اشتباه کلاس آن را B تشخیص داده است.

TN: تعداد نمونه‌هایی است که کلاس واقعی آن‌ها از نوع B بوده و روش پیشنهادی نیز به‌درستی کلاس آن را B تشخیص داده است.

حال بر اساس روابط و تعاریف فوق، معیارهای دقت، صحت، فراخوان و معیار F به شرح زیر محاسبه می‌گردد:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (۷)$$

$$Precision = \frac{TP}{TP+FP} \quad (۸)$$

$$Recall = \frac{TP}{TP+FN} \quad (۹)$$

$$F\ Score = \frac{2*Precision*Recall}{Precision+Recall} \quad (۱۰)$$

۵-۳. مجموعه داده

به‌منظور پیش‌بینی خطای نرم‌افزار از مجموعه داده‌های استاندارد بیان‌شده در مقاله [۳۰]استفاده‌شده است که مشخصات آن در جدول (۲) آورده شده است.

جدول (۲). مشخصات مجموعه داده پیش‌بینی خطای نرم‌افزار

مجموعه داده	نسخه	تعداد نمونه	تعداد نمونه خطادار	درصد نمونه خطادار
ant	1.7	745	166	0.223
camel	1.2	608	216	0.355
camel	1.4	872	145	0.166
camel	1.6	965	188	0.195
jedit	3.2	272	90	0.331
jedit	4.0	306	75	0.245
jedit	4.1	312	79	0.253
jedit	4.2	367	48	0.131
Log4j	1.0	135	34	0.252
Log4j	1.1	109	37	0.339
Log4j	1.2	205	189	0.922
lucene	2.0	195	91	0.467
lucene	2.2	247	144	0.583
lucene	2.4	340	203	0.597
xalan	2.4	723	110	0.152
xalan	2.5	803	387	0.482
xalan	2.6	885	411	0.464

۵-۴. نتایج آزمایش

در این بخش، نتایج ارزیابی روش پیشنهادی بر اساس معیارهای

دقت، صحت، فراخوان و معیار F موردبررسی قرارگرفته است. قبل از بررسی نتایج روش پیشنهادی، نتایج اسب وحشی بر روی توابع استاندارد آورده شده است که این نتایج در جدول (۳) نشان داده‌شده است. همان‌طور که مشاهده می‌شود، نتیجه ترکیب نظریه آشوب و اسب وحشی نسبت به الگوریتم اسب وحشی بهتر است.

جدول (۳). تأثیر نظریه آشوب بر روی نتایج اسب وحشی

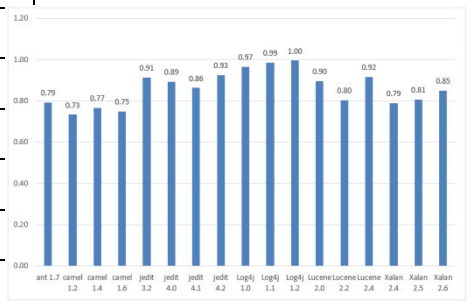
DataSet	اسب	اسب- نظریهٔ آشوب
F1	4.8 e-50	1.5 e-51
F2	4.3 e-26	5.4 e-28
F3	1.16 e-26	2.4 e-28
F4	1.7 e-16	54. e-18
F5	26.80	25.18
F6	1.9 e-04	5.18 e-05

در جدول (۴)، نتایج ارزیابی روش پیشنهادی بر روی مجموعه داده استاندارد پیش‌بینی خطای نرم‌افزار بر اساس معیارهای دقت، صحت، فراخوان و معیار F آورده شده است.

جدول (۴). نتایج ارزیابی روش پیشنهادی بر روی مجموعه داده استاندارد پیش‌بینی خطای نرم‌افزار

DataSet	Accuracy	Precision	Recall	F Measure
ant 1.7	0.7925	0.785	0.7735	0.779
camel 1.2	0.7325	0.715	0.70	0.707
camel 1.4	0.765	0.7525	0.745	0.749
camel 1.6	0.7475	0.735	0.710	0.722
jedit 3.2	0.9125	0.895	0.88	0.887
jedit 4.0	0.8925	0.8875	0.875	0.881
jedit 4.1	0.8625	0.8515	0.845	0.848
jedit 4.2	0.9250	0.895	0.8775	0.886
Log4j 1.0	0.9650	0.955	0.9475	0.951
Log4j 1.1	0.9850	0.980	0.9750	0.977
Log4j 1.2	0.9950	0.990	0.990	0.990
Lucene 2.0	0.8950	0.885	0.8775	0.881
Lucene 2.2	0.8025	0.785	0.770	0.777
Lucene 2.4	0.9150	0.905	0.89	0.897
Xalan 2.4	0.79	0.785	0.77	0.777
Xalan 2.5	0.805	0.795	0.7775	0.786
Xalan 2.6	0.85	0.835	0.825	0.830

در شکل (۱۵) مقایسه‌ای بین مجموعه داده‌های مختلف بر اساس معیار دقت نشان داده‌شده است.



شکل (۱۵). نتایج روش پیشنهادی بر روی مجموعه داده پیش‌بینی خطا بر اساس معیار دقت

همان‌طور که در شکل (۱۵) مشاهده می‌شود، بیشترین مقدار دقت مربوط به مجموعه داده *Log4j 1.2* است که این مقدار برابر ۹۹٫۵٪ است و کمترین دقت مربوط به مجموعه داده *camel 1.2* است که این مقدار برابر ۷۳٫۲۵٪ است.

۵-۵. مقایسه با نتایج کارهای مرتبط

در این بخش، نتایج ارزیابی روش پیشنهادی با نتایج مقاله [۳۰]بر اساس معیار دقت مورد مقایسه قرارگرفته است. در حالت نخست نتایج روش پیشنهادی در دو حالت انتخاب ویژگی و عدم انتخاب ویژگی مورد مقایسه قرارگرفته است.

جدول (۵). مقایسه روش انتخاب ویژگی و عدم انتخاب ویژگی

DataSet	Non Feature Selection			
	Accuracy	Precision	Recall	F Measure
ant 1.7	0.6511	0.6225	0.6430	0.6326
camel 1.2	0.6415	0.6035	0.6285	0.6157
camel 1.4	0.7145	0.7325	0.7250	0.7287
camel 1.6	0.6465	0.6325	0.6280	0.6302
jedit 3.2	0.8600	0.8435	0.8285	0.8359
jedit 4.0	0.8160	0.7935	0.7915	0.7925
jedit 4.1	0.8550	0.8425	0.8310	0.8367
jedit 4.2	0.8735	0.8625	0.8550	0.8587
Log4j 1.0	0.9275	0.9085	0.8935	0.9009
Log4j 1.1	0.9550	0.9325	0.9415	0.9370
Log4j 1.2	0.9630	0.9525	0.9600	0.9562
Lucene 2.0	0.8345	0.8215	0.7960	0.8085
Lucene 2.2	0.7615	0.7545	0.7380	0.7462
Lucene 2.4	0.8950	0.8670	0.8830	0.8749
Xalan 2.4	0.7160	0.7035	0.6850	0.6941
Xalan 2.5	0.7435	0.7260	0.7150	0.7205
Xalan 2.6	0.8320	0.8015	0.7945	0.7980
DataSet	Feature Selection			
	Accuracy	Precision	Recall	F Measure
ant 1.7	0.7925	0.785	0.7735	0.779
camel 1.2	0.7325	0.715	0.70	0.707
camel 1.4	0.765	0.7525	0.745	0.749
camel 1.6	0.7475	0.735	0.710	0.722
jedit 3.2	0.9125	0.895	0.88	0.887
jedit 4.0	0.8925	0.8875	0.875	0.881
jedit 4.1	0.8625	0.8515	0.845	0.848
jedit 4.2	0.9250	0.895	0.8775	0.886
Log4j 1.0	0.9650	0.955	0.9475	0.951
Log4j 1.1	0.9850	0.980	0.9750	0.977
Log4j 1.2	0.9950	0.990	0.990	0.990
Lucene 2.0	0.8950	0.885	0.8775	0.881
Lucene 2.2	0.8025	0.785	0.770	0.777
Lucene 2.4	0.9150	0.905	0.89	0.897
Xalan 2.4	0.79	0.785	0.77	0.777
Xalan 2.5	0.805	0.795	0.7775	0.786
Xalan 2.6	0.85	0.835	0.825	0.830

باتوجه‌به نتایج جدول (۵)، روش انتخاب ویژگی در پیش‌بینی خطای نرم‌افزار به دلیل مؤثر بودن و کارایی بیشتر نسبت به روش عدم انتخاب ویژگی، ترجیح داده می‌شود. انتخاب ویژگی به معنای انتخاب ویژگی‌های مهم و مؤثر در پیش‌بینی خطاهای نرم‌افزار است. این روش به ما امکان می‌دهد تا ویژگی‌هایی که

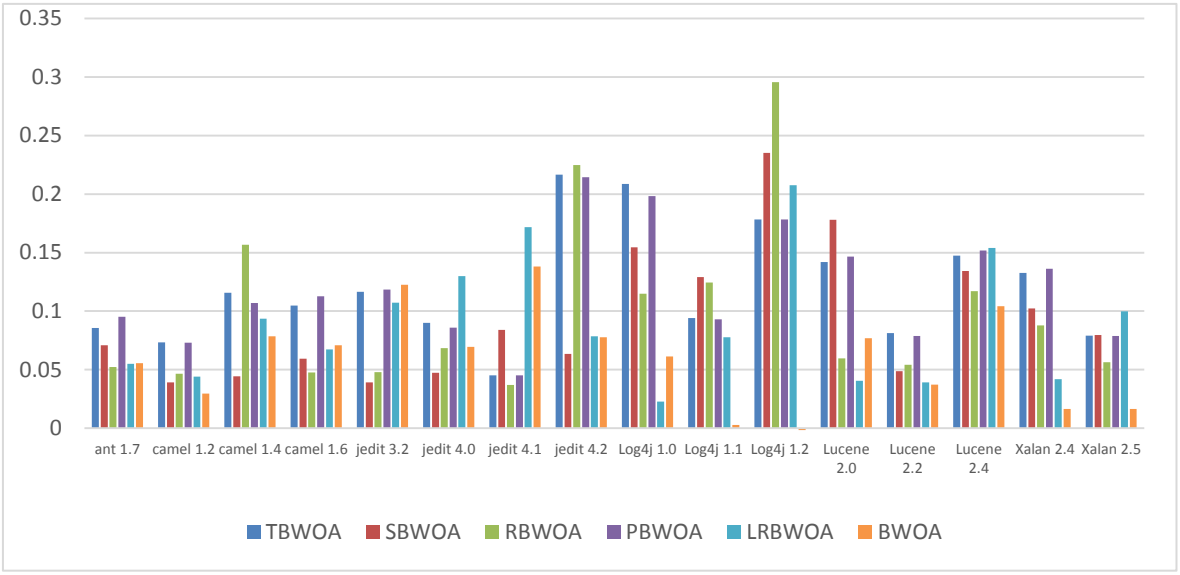
بیشترین تأثیر را در پیش‌بینی خطا دارند را شناسایی کنیم و بر روی آن‌ها تمرکز کنیم. انتخاب ویژگی به ما کمک می‌کند تا مجموعه داده‌ها را کاهش داده و به‌جای استفاده از تمام ویژگی‌ها، فقط از ویژگی‌های مهم استفاده کنیم. این کاهش ابعاد داده‌ها باعث افزایش کارایی مدل پیش‌بینی می‌شود و به ما امکان می‌دهد تا بادقت بیشتری خطاهای نرم‌افزار را پیش‌بینی کنیم.

به‌طورکلی، روش انتخاب ویژگی نسبت به روش عدم انتخاب ویژگی در پیش‌بینی خطای نرم‌افزار مؤثرتر است زیرا باعث بهبود دقت و کارایی مدل پیش‌بینی شده و زمان و منابع مورد نیاز برای آن را کاهش می‌دهد. در گام بعدی، از الگوریتم طبقه‌بندی درخت تصمیم استفاده‌شده است. نتایج این آزمایش در جدول (۶) نشان داده‌شده است.

جدول (۶). مقایسه نتایج روش پیشنهادی با نتایج مقاله [۳۰]

DataSet	روش پیشنهادی	مرجع [۳۰]					
	HOA	BWOA	LRBWOA	PBWOA	RBWOA	SBWOA	TBWOA
ant 1.7	**0.7925**	0.7069	0.7216	0.7402	0.6974	0.7375	0.7369
camel 1.2	0.7325	0.6591	0.6933	0.6860	0.6595	0.6885	0.7029
camel 1.4	0.765	0.6493	0.7207	0.6083	0.6581	0.6714	0.6864
camel 1.6	0.7475	0.6426	0.6881	0.6999	0.6347	0.6803	0.6767
jedit 3.2	0.9125	0.7959	0.8734	0.8647	0.7942	0.8054	0.7899
jedit 4.0	0.8925	0.8024	0.8450	0.8242	0.8067	0.7625	0.8229
jedit 4.1	0.8625	0.8173	0.7786	0.8256	0.8173	0.6909	0.7245
jedit 4.2	0.9250	0.7085	0.8615	0.7003	0.7107	0.8466	0.8472
Log4j 1.0	0.9650	0.7565	0.8104	0.8500	0.7667	0.9421	0.9036
Log4j 1.1	0.9850	0.8908	0.8559	0.8606	0.8920	0.9073	0.9822
Log4j 1.2	0.9950	0.8167	0.7598	0.6995	0.8167	0.7875	1.00
Lucene 2.0	0.8950	0.7530	0.7171	0.8353	0.7485	0.8545	0.8180
Lucene 2.2	0.8025	0.7212	0.7538	0.7482	0.7236	0.7634	0.7653
Lucene 2.4	0.9150	0.7675	0.7807	0.7980	0.7633	0.7609	0.8108
Xalan 2.4	0.79	0.6573	0.6878	0.7023	0.6539	0.7482	0.7735
Xalan 2.5	0.805	0.7260	0.7255	0.7485	0.7261	0.7051	0.7884

همان‌طور که در جدول (۶) مشاهده می‌شود، کارایی الگوریتم اسب وحشی (درروش پیشنهادی) نسبت به الگوریتم وال (در مقاله [۳۰]) جهت انتخاب ویژگی، برتر است، زیرا الگوریتم اسب وحشی توانسته است فضای مساله را به‌صورت مناسب پیمایش نماید. همچنین، برتری روش پیشنهادی به دلیل استفاده از نظریه آشوب است که از تولید جمعیت اولیه تصادفی اجتناب شده و تنوع جواب‌ها حفظ شده است. در شکل (۱۶) میزان برتری روش پیشنهادی نسبت به دیگر روش‌ها نشان داده‌شده است. جهت محاسبه برتری روش پیشنهادی نسبت به نتایج مرجع [۳۰]، از اختلاف دو روش استفاده‌شده است.



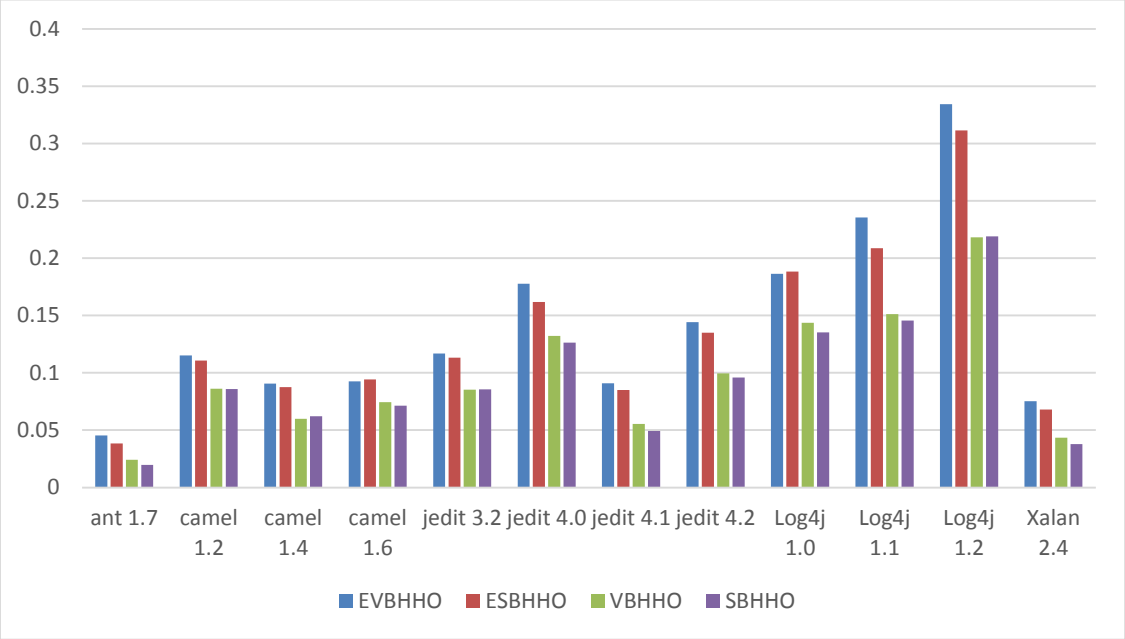
شکل (۱۶). نشان‌دهنده برتری روش پیشنهادی با روش‌های مرجع [۳۰]

در جدول (۷) مقایسه بین روش پیشنهادی و مقاله [۲۹] مورد مقایسه قرار گرفته است.

جدول (۷). مقایسه نتایج روش پیشنهادی با نتایج مقاله [۲۹]

DataSet	روش پیشنهادی	مرجع [۲۹]			
	HOA	SBHHO	VBHHO	ESBHHO	EVBHHO
ant 1.7	0.7925	0.7471	0.7541	0.7683	0.7727
camel 1.2	0.7325	0.6174	0.6217	0.6464	0.6467
camel 1.4	0.765	0.6745	0.6776	0.7051	0.7029
camel 1.6	0.7475	0.6549	0.6532	0.6731	0.6762
jedit 3.2	0.9125	0.7958	0.7994	0.8271	0.8270
jedit 4.0	0.8925	0.7149	0.7308	0.7602	0.7661
jedit 4.1	0.8625	0.7715	0.7775	0.8071	0.8133
jedit 4.2	0.9250	0.7808	0.7901	0.8256	0.8290
Log4j 1.0	0.9650	0.7786	0.7766	0.8214	0.8297
Log4j 1.1	0.9850	0.7495	0.7762	0.8337	0.8395
Log4j 1.2	0.9950	0.6606	0.6837	0.7767	0.7761
Xalan 2.4	0.79	0.7149	0.7220	0.7467	0.7521

همان‌طور که در جدول (۷) مشاهده می‌شود، مقایسه‌ای بین روش پیشنهادی و نتایج مرجع [۲۹] انجام‌شده است. در این مرجع، عملیات پیش‌بینی بر اساس الگوریتم شاهین هریس انجام‌شده است که نتایج روش پیشنهادی نسبت به آن افزایش یافته است، زیرا الگوریتم شاهین هریس در بهینگی محلی قرار می‌گیرد. در شکل (۱۷)، میزان برتری روش پیشنهادی نسبت به مرجع [۲۹] آورده شده است.



شکل (۱۷). میزان برتری روش پیشنهادی با روش‌های مرجع [۲۹]

۶. نتیجه‌گیری و کارهای آینده

در این مقاله، جهت عملیات انتخاب ویژگی از روش‌های پوشش استفاده‌شده است. الگوریتم مورداستفاده در این مقاله برای انتخاب ویژگی، الگوریتم اسب وحشی است. مهم‌ترین ضعف الگوریتم‌های تکاملی، قرارگرفتن آنها در بهینگی محلی است. برای رفع این مشکل از نظریه آشوب استفاده‌شده است. درنهایت، جهت پیش‌بینی خطای نرم‌افزار از روش‌های یادگیری ماشین شامل درخت تصمیم استفاده‌شده است. همچنین، جهت پیش‌بینی خطای نرم‌افزار از مجموعه داده‌های استاندارد بیان‌شده در مقاله [۳۰]استفاده‌شده است. نتایج ارزیابی روش پیشنهادی بر اساس معیارهای دقت، صحت، فراخوان و معیار F نشان می‌دهد که بیشترین مقدار دقت مربوط به مجموعه‌داده *Log4j 1.2* است که این مقدار برابر ۹۹,۵٪ است و کمترین دقت مربوط به مجموعه‌داده *camel 1.2* است که این مقدار برابر ۷۳,۲۵٪ است. درنهایت، مقایسه روش پیشنهادی و روش ارائه‌شده در مقاله [۳۰]بر اساس معیار دقت، نشان می‌دهد که کارایی الگوریتم اسب وحشی (درروش پیشنهادی) نسبت به الگوریتم وال (در مقاله [۳۰]) جهت انتخاب ویژگی بهتر است. زیرا الگوریتم اسب وحشی

توانسته است فضای مسئله را به‌صورت مناسب پیمایش نماید. از طرفی، درروش پیشنهادی از نظریه آشوب برای اجتناب از تولید جمعیت اولیه تصادفی و ایجاد تنوع جواب‌ها استفاده‌شده است که این امر، در برتری و کارایی روش پیشنهادی مؤثر بوده است.

برای توسعه و تقویت روش پیشنهادی در آینده، راهکارهای زیر پیشنهاد می‌شود:

بهبود الگوریتم انتخاب ویژگی با دیگر الگوریتم‌های تکاملی نظیر الگوریتم تکاملی پلیکان.

استفاده از ترکیب روش‌های فیلتر و پوشش جهت بهبود روش انتخاب ویژگی و افزایش کارایی مدل پیش‌بینی خطا.

استفاده از ترکیب مدل‌های یادگیری ماشین از قبیل جنگل تصادفی، ژنتیک و ازدحام ذرات.

۷. مراجع

[1] S. S. Rathore and S. Kumar, "A study on software fault prediction techniques," Artificial Intelligence Review, vol. 51, no. 2, pp. 255-327, Feb. 2019. https://doi.org/10.1007/s10462-017-9563-5.

[2] W. Rhmann, B. Pandey, G. Ansari, and D. K. Pandey, "Software fault prediction based on change metrics using hybrid algorithms: An empirical study," Journal of King Saud University - Computer and Information Sciences, vol. 32, no. 4, pp. 419-424, May 2020. https://doi.org/10.1016/j.jksuci.2019.03.006.

[3] S. S. Rathore and S. Kumar, "An empirical study of ensemble techniques for software fault prediction," Applied Intelligence, vol. 51, no. 6, pp. 3615-3644, Jun. 2021. https://doi.org/10.1007/s10489-020-01935-6.

[4] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Systems with Applications, vol. 42, no. 4, pp. 1872-1879, 2015. https://doi.org/10.1016/j.eswa.2014.10.025.

[5] D. Sharma and P. Chandra, "A comparative analysis of soft computing techniques in software fault prediction model development," International Journal of Information Technology, vol. 11, no. 1, pp. 37-46, Mar. 2019. https://doi.org/10.1007/s41870-018-0211-3.

[6] V. Garousi, M. Felderer, and F. N. Kılıçaslan, "A survey on software testability," Information and Software Technology, vol. 108, pp. 35-64, 2019. https://doi.org/10.1016/j.infsof.2018.12.003.

[7] B. J. Akinsanya et al., "Machine Learning and Value Generation in Software Development: A Survey," Communications in Computer and Information Science, vol. 1288 CCIS, pp. 44-55, 2021. https://doi.org/10.1007/978-3-030-71472-7_3.

[8] S. Masuda, K. Ono, T. Yasue, and N. Hosokawa, "A survey of software quality for machine learning applications," in Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018, pp. 279-284, 2018. 10.1109/ICSTW.2018.00061.

[9] J. Chen, V. Nair, and T. Menzies, "Beyond evolutionary algorithms for search-based software engineering," arXiv preprint arXiv:1701.07950,2017.https://doi.org/10.1016/j.infsof.2017.08.007.

[10] S. S. Rathore and S. Kumar, "Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems," Knowledge-Based Systems, vol. 119, pp. 232-256, 2017. https://doi.org/10.1016/j.knosys.2016.12.017.

[11] A. Karimi and F. Karimi, "A method to prediction of software system's code smells using neural network," 1402. Scientific Journal of Electronic & Cyber Defense Vol. 11, No. 3, autumn 2023, Serial No. 43ISSN: 2322-4347, E-ISSN: 2980-8979 (In Persian). https://civilica.com/doc/1918388.

[12] S. Bejani and A. H. Kay Manesh, "A novel way to identify effective test-case in software testing," 1402. Scientific Journal of Electronical & Cyber Defence Vol. 11, No. 2, summer 2023, Serial No. 42 (In Persian). https://civilica.com/doc/1756097.

[13] Wahono, R.S., "A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks," Journal of Software Engineering, vol. 1, pp. 1-16, 2017. http://journal.ilmukomputer.org.

[14] M. Ishraghinia, A. Karimi, and I. Bastami, "A model for Feature Selection in Software Fault Prediction Based on the Memetic Algorithm and Fuzzy Logic," 1400. Journal of Electronical & Cyber Defence Vol. 9, No. 3, 2021, Serial No. 35, (In Persian). https://civilica.com/doc/1346724

[15] B. Boehm and V. R. Basili, "Top 10 list [software development]," Computer, vol. 34, no. 1, pp. 135-137, 2011. doi:10.1109/2.962984.

[16] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: current results, limitations, new approaches," Automated Software Engineering, vol. 17, no. 4, pp. 375-407, 2010. https://doi.org/10.1007/s10515-010-0069-5.

[17] S. Omri and C. Sinz, "Deep Learning for Software Defect Prediction," in Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, New York, NY, USA: ACM, 2020, pp. 209-214. https://doi.org/10.1063/5.0178913.

[18] M. Ravat and S. Dubey, "Software defect prediction models for quality improvement: a literature study," *Int. J. Comput. Sci. Issues*, vol. 9, pp. 288-296, 2012.

[18] W. Huang and C. Chieh-Jen, "A GA-based feature selection and parameters optimization for support vector machines," *Expert Syst. Appl.*, vol. 31, no. 2, pp. 231-240, 2006. https://doi.org/10.1016/j.eswa.2005.09.024.

[19] M. A. Tahir, A. Bouridane, and F. Kurugollu, "Simultaneous feature selection and feature weighting using hybrid tabu search/k-nearest neighbor classifier," Pattern Recognition Letters, vol. 28, no. 4, pp. 438–446, 2007. https://doi.org/10.1016/j.patrec.2006.08.016.

[20] B. Zarei and M. R. Meybodi, "Improving learning ability of learning automata using chaos theory," Journal of Supercomputing, vol. 77, no. 1, pp. 652–678, 2021. https://doi.org/10.1007/s11227-020-03293-z.

[21] M. Sharafi, F. Fotouhi-Ghazvini, M. Shirali, and M. Ghassemian, "A low power cryptography solution based on chaos theory in wireless sensor nodes," IEEE Access, vol. 7, pp. 8737–8753, 2019. 10.1109/ACCESS.2018.2886384.

[22] A. M. Anter and M. Ali, "Feature selection strategy based on hybrid crow search optimization algorithm integrated with chaos theory and fuzzy c-means algorithm for medical diagnosis problems," Soft Computing, vol. 24, no. 3, pp. 1565–1584, 2020. https://doi.org/10.1007/s00500-019-03988-3.

[23] G. Martin, S. Kaviraj, A. Hocking, S.C. Read, and J.E. Geach, "Galaxy morphological classification in deep-wide surveys via unsupervised machine learning," Monthly Notices of the Royal Astronomical Society, vol. 491, no. 1, pp. 1408–1426, 2020. https://doi.org/10.1093/mnras/stz3006.

[24] A.C. Lorena, L.P.F. Garcia, J. Lehmann, M.C.P. Souto, and

T.K.A.M. Ho, "How complex is your classification problem? A survey on measuring classification complexity," ACM Computing Surveys, vol. 52, no. 5, 2019. https://doi.org/10.1145/3347711.

[25] M. Bazoband and H. Bahramgiri, "A New Hybrid Approach for Traffic Identification and Classification in Wireless Networks," 1401. Scientific Journal of Electronical & Cyber Defence Vol. 10, No. 2, summer 2022, Serial No. 38. (In Persian). https://civilica.com/doc/1602313.

[26] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," Journal of Applied Science and Technology Trends, vol. 2, no. 01, pp. 20-28, 2021. https://doi.org/10.38094/jastt20165.

[27] J. Mrva, Š. Neupauer, L. Hudec, J. Ševcech, and P. Kapec, "Decision Support in Medical Data Using 3D Decision Tree Visualisation," in 2019 E-Health and Bioengineering Conference (EHB), Nov. 2019, pp. 1-4. 10.1109/EHB47216.2019.8969926.

[28] I. Tumar, Y. Hassouneh, H. Turabieh, and T. Thaher, "Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction," IEEE Access, vol. 8, pp. 8041–8055, 2020. 10.1109/ACCESS.2020.2964321.

[29] T. Thaher and N. Arman, "Efficient Multi-Swarm Binary Harris Hawks Optimization as a Feature Selection Approach for Software Fault Prediction," in 2020 11th International Conference on Information and Communication Systems, ICICS 2020, pp. 249–254, 2020. 10.1109/ICICS49469.2020.239557.

[30] Y. Hassouneh, H. Turabieh, T. Thaher, I. Tumar, H. Chantar, and J. Too, "Boosted Whale Optimization Algorithm with Natural Selection Operators for Software Fault Prediction," IEEE Access, vol. 9, pp. 14239–14258, 2021. 10.1109/ACCESS.2021.3052149.

[31] H. Alsghaier and M. Akour, "Software fault prediction using Whale algorithm with genetics algorithm," Softw. Pract. Exp., vol. 51, no. 5, pp. 1121–1146, May 2022. https://doi.org/10.1002/spe.2941.

[32] M. R. Ahmed, M. A. Ali, N. Ahmed, M. F. Bin Zamal, and F. M. J. M. Shamrat, "The Impact of Software Fault Prediction in Real-World Application," in Proceedings of 2020 the 6th International Conference on Computing and Data Engineering, 2020, pp. 247–251. https://doi.org/10.1145/3379247.3379278.

[33] O. Al Qasem, M. Akour, and M. Alenezi, "The Influence of Deep Learning Algorithms Factors in Software Fault Prediction," IEEE Access, vol. 8, pp. 63945–63960, 2020. 10.1109/ACCESS.2020.2985290.

[34] F. MiarNaeimi, G. Azizyan, and M. Rashki, "Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems," Knowledge-Based Syst., vol. 213,p.106711,Feb.2021.https://doi.org/10.1016/j.knosys.2020.106711.