علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۲۳، شماره ۱، ۱۴۰۴
صفحات ۳۱-۴۵
مقاله پژوهشی

پیش‌بینی خطاهای نرم‌افزاری با دانه‌بندی داده‌ها

بهروز شاهی[۱]، هومان تحیری[۲]*
*نویسنده مسئول
دریافت: ۱۴۰۳/۰۷/۰۷، بازنگری: ۱۴۰۳/۰۹/۱۰، پذیرش: ۱۴۰۴/۰۱/۲۵

[۱] دانشجوی دکتری مهندسی نرم افزار، بخش مهندسی و علوم کامپیوتر و فناوری اطلاعات، دانشگاه شیراز، شیراز، ایران
b.shahi@shirazu.ac.ir
[۲] دانشیار، بخش مهندسی و علوم کامپیوتر و فناوری اطلاعات ، دانشگاه شیراز، شیراز، ایران
tahayori@shirazu.ac.ir

چکیده

به دلیل پیچیدگی بالای سیستم‌های نرم‌افزاری و با توجه به محدودیت زمان و هزینه توسعه ، کشف تمامی خطاهای نرم افزار در زمان توسعه، بسیار دشوار می‌باشد. بنابراین پیش‌بینی خطاها با استفاده از نسخه‌های قبلی پروژه‌ها یا پروژه‌های دیگر جهت ایجاد نرم‌افزار با قابلیت اطمینان بالا ضروری است. عدم تعمیم‌پذیری الگوریتم‌های پیش‌بینی خطاهای نرم‌افزاری، عدم تعادل بین داده‌های خطا و بدون خطا و متفاوت بودن مقادیر ویژگی‌ها در پروژه‌های مختلف، چالش‌های اساسی در سیستم‌های پیش‌بینی خطاهای نرم‌افزار می‌باشند. این مقاله یک معماری چهار سطحی برای پیش‌بینی خطاهای نرم‌افزاری ارایه می‌کند. ابتدا الگوریتمی برای دانه بندی مقادیر عددی معرفی می‌شود. سپس اهمیت ویژگی‌ها بر اساس مقادیر دانه بندی شده محاسبه می‌شود. سپس داده‌ها به سه قسمت آموزشی، اعتبارسنجی و تست تقسیم می‌شود و برای متعادل کردن نسبت داده‌های خطا به داده‌های غیرخطا بر روی داده‌های آموزشی از الگوریتم oversampling استفاده می‌شود. در انتها سه نوع ارزیابی و مقایسه با سایر روش‌ها با استفاده از دیتاست‌های PROMISE و NASA انجام می‌پذیرد. نتایج این مقاله نشان می‌دهد که پیش‌بینی خطا در هر سه نوع ارزیابی در اکثر پروژه‌های هر یک از دیتاست‌ها عملکرد بهتری نسبت به سایر روش‌ها دارد. همچنین نشان داده می شود که تعمیم پذیری ارایه شده در این مقاله بر روی هر مجموعه داده‌ای قابل انجام است.

کلمات کلیدی: قابلیت اطمینان نرم‌افزار، خطاهای نرم‌افزار، دانه بندی اطلاعات، تعمیم‌پذیری، یادگیری ماشین

۱- مقدمه

با توجه به نقش روزافزون نرم‌افزارها در تمامی امور و اهمیت کارکرد صحیح آنها، لزوم به کارگیری روش‌ها و اصول مهندسی نرم‌افزار در مراحل توسعه، مدیریت و پشتیبانی آنها بیشتر نمود می‌یابد. اهمیت موضوع کیفیت نرم‌افزار در تولید نرم‌افزارهای مشتری‌مدار بسیار بالاست و توجه دقیق به همین امر می‌تواند منجر به ایجاد نرم‌افزارهای قدرتمند گردد. فرآیند توسعه نرم‌افزار می‌بایست متضمن تولید یک سیستم با ویژگی‌های خاص و خواسته شده باشد. این فرایند از مرحله طرح یک راه حل معین برای مسئله خواسته شده (امکان سنجی) آغاز می‌شود. در واقع، پس از دریافت خواسته و تحلیل سیستم، طراحی صورت گرفته و در نهایت این طراحی با کمک ابزارهای پیاده سازی به یک سیستم واقعی تبدیل می شود. این فرایند می‌بایست از یک سو متضمن بر آورده ساختن نیازهای کاربران و از سوی دیگر تضمین کننده کیفیت و عملکرد مناسب سیستم باشد. مرحله بعدی در توسعه نرم‌افزار، مرحله تست (آزمایش) و رفع اشکالات نرم‌افزار است که هدف آن ارتقا کیفیت محصول ساخته شده می‌باشد.

در چند دهه اخیر پیچیدگی نرم‌افزار باعث ایجاد دوران جدیدی در زمینه دانش تولید نرم‌افزار شده است. خواسته‌های مشتریان و لزوم راحتی استفاده از سیستم‌های نرم‌افزاری، پیچیدگی تولید نرم‌افزار را بالا برده است و با افزایش

پیچیدگی‌ها میزان خطاهای نرم‌افزاری نیز افزایش یافته است. با در نظر گرفتن این حقیقت که حدوداً ۷۵٪ چرخه حیات به تست (آزمایش) و نگهداری نرم‌افزار سپری می‌شود [۱]، مسئله یافتن خطا اهمیت ویژه‌ای پیدا کرده است. پیچیدگی تست‌ها (آزمایش‌ها) و سپس رفع اشکالات، عامل افزایش سریع هزینه‌های توسعه و نگهداری می‌باشد که البته تاثیر بسزایی نیز در ارتقاء کیفیت و بهره‌وری محصول دارد. از این رو است که ، روش‌های پیش‌بینی خطای نرم-افزار جهت کاهش هزینه‌ها مطرح شده است [۲-۴].

پیش بینی خطاهای نرم‌افزاری دارای سه چالش اساسی می‌باشد. اولا، عدم تعمیم پذیری الگوریتم‌های ارایه شده برای پیش‌بینی خطاهای نرم‌افزاری باعث شده است که نتایج الگوریتم‌ها بر روی مجموعه داده‌های دیگر نتایج خوبی نداشته باشند. عدم تعمیم پذیری همچنین باعث می‌شود که شرکت‌های نوپا نتوانند از تجربیات و دیتاست‌های دیگران بخوبی استفاده کنند تا بتوانند در کمترین زمان محصول خود را روانه بازار کنند. ثانیا، به دلیل متغیر بودن بازه-های داده‌های عددی در مجموعه‌های داده مرتبط با خطای نرم‌افزار به راحتی نمی‌توان ویژگی‌های مهم را محاسبه کرد. ثالثا، به دلیل عدم تعادل داده‌ها بین کلاس‌های خطا و بدون خطا، بیشتر الگوریتم‌های طبقه‌بندی نتایج خوبی تولید نمی‌کنند و الگوریتم‌های ارایه شده بر یک مجموعه داده، برای سایر مجموعه داده‌ها کارایی ندارند.

مهم‌ترین ویژگی که این مقاله ارایه می‌دهد تعمیم‌پذیر بودن روش ارایه شده می‌باشد. ما در این مقاله با دانه‌بندی و یافتن ویژگی‌های مهم هم عملکرد سیستم را بالا می‌بریم و هم نشان می‌دهیم که روش ارایه شده بر روی هر مجموعه داده قابل انجام می‌باشد. در حالی که بیشتر روش‌ها [۵,۳۲,۳۳] فقط بر روی یک مجموعه داده کارایی دارند. همچنین ممکن است نتایج آنها بر روی مجموعه داده دیگر نتایج خوبی تولید نکند. الگوریتم دانه‌بندی ارایه شده در این مقاله به دلیل مقاوم بودن در برابر داده‌های پرت باعث ایجاد بازه‌های مناسب جهت تبدیل اعداد به کلمات LOW، MEDIUM و HIGH و بهبود پارامترهای ارزیابی دقت و AUC و F-measure می‌گردد. همچنین الگوریتم دانه‌بندی باعث کاهش تنوع مقادیر ویژگی‌ها می‌شود که خود عامل بسیار مهم در تعمیم-پذیری و قابلیت استفاده مجدد روش ارایه شده برای مجموعه داده مختلف می‌گردد. در حالی که در روش‌های دیگر همچون [۵] به دلیل استفاده از میانگین و نامقاوم بودن در برابر داده‌های پرت نتایج مطلوبی تولید نمی‌کند.

راهکار ارایه شده در این مقاله مبتنی بر دانه‌بندی[1] داده‌های عددی ، و تعیین اهمیت ویژگی‌های بدست آمده می باشد. تعمیم پذیری الگوریتم به دلیل درنظر گرفته شدن مقادیر دانه‌بندی شده، به راحتی انجام‌پذیر می‌شود. روش پیشنهادی به این صورت است که ابتدا راهکاری برای دانه‌بندی داده‌های عددی مطرح می‌شود که طی آن مقادیر هر یک از ویژگی‌های نرم‌افزار به بازه تبدیل می‌شود. پس از حصول بازه‌ها، با استفاده از روش‌های Information Gain و Chi Squared اهمیت هر ویژگی بدست می‌آید که برای هر ویژگی امتیازی مشخص می‌گردد. بجای استفاده از تمامی ویژگی‌ها در طبقه‌بندی، از پنج ویژگی با امتیاز بالا استفاده می‌شود. سپس به دلیل مسئله عدم تناسب بین کلاس‌های بدون خطا و خطادار، با روش oversampling تناسب و تعادل بر روی داده‌های آموزشی ارایه می‌شود. با رفع مساله عدم تعادل داده‌ها، کارایی الگوریتم‌های طبقه‌بندی تضمین می‌شود. در نهایت برای ارزیابی روش پیشنهادی از دیتاست‌های PROMISE و NASA استفاده می‌شود. نتایج حاصل، نشان می‌دهد که ویژگی‌های با امتیاز بالا در طبقه‌بندی و پیش بینی خطاها عملکرد بهتری دارند.

این مقاله به صورت زیر سازماندهی شده است. در بخش دوم کارهای مرتبط بررسی می‌شود. در بخش سوم راه‌کار ارائه شده مطرح می‌گردد. بخش چهارم به بررسی نتایج و آزمایش راه‌کار ارائه شده تخصیص یافته است و در بخش پنجم نتیجه‌گیری و کارهای آتی ارایه شده است.

۲- کارهای مرتبط

در زمینه پیش‌بینی خطاهای نرم‌افزاری تحقیقات زیادی انجام شده‌است. این کارها را می توان به سه دسته تقسیم کرد. برخی از این روش‌ها برای پیش‌بینی تعداد خطاهای نرم‌افزار از الگوریتم‌های یادگیری ماشین بهره می‌برند. برخی دیگر برای پیش‌بینی تعداد خطاها از الگوریتم‌های فازی استفاده می‌کنند و دسته سوم از روش ترکیبی فازی و یادگیری ماشین استفاده می‌کنند [۵-۷]. در این مقاله نیز از روش ترکیبی استفاده می‌شود.

در روش‌های مبتنی بر تکنیک‌های یادگیری ماشینی برای بهبود نرخ پیش‌بینی خطا در سیستم‌های نرم‌افزاری از رگرسیون لجستیک، Bayes Naïve و الگوریتم‌های یادگیری ماشین مانند درخت تصمیم، پرسپترون و SVM، برای پیش‌بینی خطاهای نرم‌افزار استفاده شده است [۸-۱۰]. در مرجع [۱۱] یک روش ساده بیز برای پیش‌بینی خطاهای نرم‌افزار پیشنهاد شده است که درآن از مجموعه داده‌های ناسا برای ارزیابی روش پیشنهادی استفاده شده است. این مطالعه نشان می‌دهد که Weighted Naïve Bayes عموماً نتایج بهتری نسبت به Naïve Bayes ایجاد می‌کند. این مقاله همچنین نشان می‌دهد که ویژگی‌های استفاده شده برای پیش‌بینی خطای نرم افزار نباید وزن یکسانی داشته باشند. به همین دلیل در بیشتر مواقع روش بیزین ساده که استقلال و وزن یکسان برای ویژگی‌ها در نظر می‌گیرد دقت کمتری برای پیش بینی خطاهای نرم‌افزار دارد. در مقاله [۱۲]، بیز ساده، رگرسیون لجستیک و شبکه بیز برای ساخت مدل پیش بینی خطا استفاده شده است. در این روش مجموعه داده‌های کدنویسی پروژه های صنعتی یک شرکت در بازه ۱۱ ماهه مورد بررسی قرار گرفته است. این مقاله از ویژگی‌های کدها مثل تعداد خط کد یا تعداد توابع و غیره در طبقه بندی استفاده کرده است. در این مقاله نیز نشان داده شده که همه ویژگی‌ها دارای اثر یکسانی برای پیش‌بینی خطای نرم‌افزار نیستند. در مقالات [۱۳-۱۶] از الگوریتم بیزین برای پیش‌بینی تعداد خطاها استفاده شده است. نتایج این مراجع نشان می‌دهد که این روش بر مجموعه داده‌های کوچک عملکرد خوبی دارد. الگوریتم‌های مبتنی بر شبکه‌های بیزین با در نظر گرفتن مستقل بودن ویژگی‌ها و راحتی محاسبه احتمالات بیزین بین ویژگی‌ها بیشتر استفاده شده اند. در مقاله [۱۷] روش‌های مختلف انتخاب ویژگی‌ها در مجموعه داده های PROMISE بررسی شده‌اند. این مقاله نشان می دهد که استفاده از متریک‌های مبتنی بر کد و الگوریتم Random Forest نتایج بهتری دارد. همچنین در [۱۸] نشان داده شده که استفاده از پرسپترون چند لایه بر روی مجموعه داده PROMISE می تواند دقت بالایی ارایه دهد. روش‌های دیگری مانند ANN در [۲۰,۱۹] برای پیش‌بینی تعداد خطاهای نرم‌افزار در پروژه‌های مختلف ارائه شده‌اند. کارهای بررسی شده تا سال ۲۰۱۸ معمولا از روش‌های

[1] Granulation

ساده همچون روش‌های مبتنی بر بیزین و درخت تصادفی و درخت تصمیم بر اساس گزارش ارایه شده در [۲۱] استفاده کرده‌اند. ویژگی‌های مورد استفاده در این الگوریتم‌ها معمولا تعداد خط برنامه و تعداد توابع و تعداد عملگرها و عملوندها می‌باشد.

دلیل اصلی استفاده از مفاهیم منطق فازی در مسایل پیش‌بینی خطای نرم‌افزار، به تحقیق انجام یافته توسط فنتون و همکارانش [۲۲] برمی‌گردد. در این مقاله برای ۳۱ پروژه، مجموعه‌ای از مقادیر کیفی با استفاده از نظرات مدیران پروژه و کارشناسان و مشتریان جمع‌آوری شد. این دیتاست برای فازهای توسعه نرم-افزار شامل فاز نیازمندی، طراحی، کدنویسی و تست جمع‌آوری شده است. در این مقاله ۲۷ ویژگی با مقادیر فازی LOW، MEDIUM و HIGH مقدار دهی شده اند. سپس در مقاله، خطا برای هر کدام از ۳۱ پروژه این دیتاست پیش بینی شده و دقت روش بررسی گردیده است. تحقیقات دیگر بر اساس این دیتاست ، شکل گرفته است. روش‌های ارایه شده در مراجع [۲۳-۲۶] سعی در بهبود دقت ارایه شده در [۲۲] داشته‌اند. معمولا هر کدام از این روش‌ها با اعمال تغییرات در فازهای متفاوت و با استفاده از فرمول‌های متفاوت مانند پروفایل فازی یا استفاده از توابع min-max فازی، توانسته‌اند دقت ارایه شده نسبت به [۲۲] را بهبود ببخشند. مشکل اصلی این مقالات اندازه کوچک مجموعه داده مورد استفاده و عدم وجود روش‌های اعتبارسنجی برای سایر داده‌ها می‌باشد. بعبارت دیگر می‌توان گفت که این روش‌ها قابلیت تعمیم ندارند.

الگوریتم‌های دیگری برای پیش‌بینی خطاهای نرم‌افزار با استفاده از ترکیبی از روش‌های یادگیری ماشین و محاسبات نرم معرفی شده‌اند [۲۷-۲۹]. در این روش‌ها معمولاً داده‌های عددی با سیستم‌های فازی به اعداد فازی تبدیل می‌شوند سپس با استفاده از الگوریتم‌های یادگیری ماشین تعداد خطاها پیش‌بینی می‌شود. کارهای ارایه شده در [۵] نیز در همین دسته جای می گیرند. روش کار این به این صورت است که ابتدا داده‌های عددی به اعداد فازی تبدیل می‌شوند که این تبدیل معمولا با درنظر گرفتن ویژگی‌های آماری مثل میانگین و میانه و چارک انجام می‌شود. سپس با استفاده از الگوریتم‌های هوش مصنوعی مثل الگوریتم ژنتیک سعی در ایجاد قوانین فازی یا انتخاب ویژگی‌ها می‌شود. در این دسته از مقالات اغلب از مجموعه داده های PROMISE و NASA استفاده شده است.

در روش ارایه شده در [۳۲] ابتدا یک الگوریتم برای حل مساله عدم تعادل داده‌ها ارایه شده است و نشان داده شده است که با بهبود این مساله بر روی مجموعه داده NASA می توان دقت پیش‌بینی خطاهای نرم‌افزاری را بالا برد.

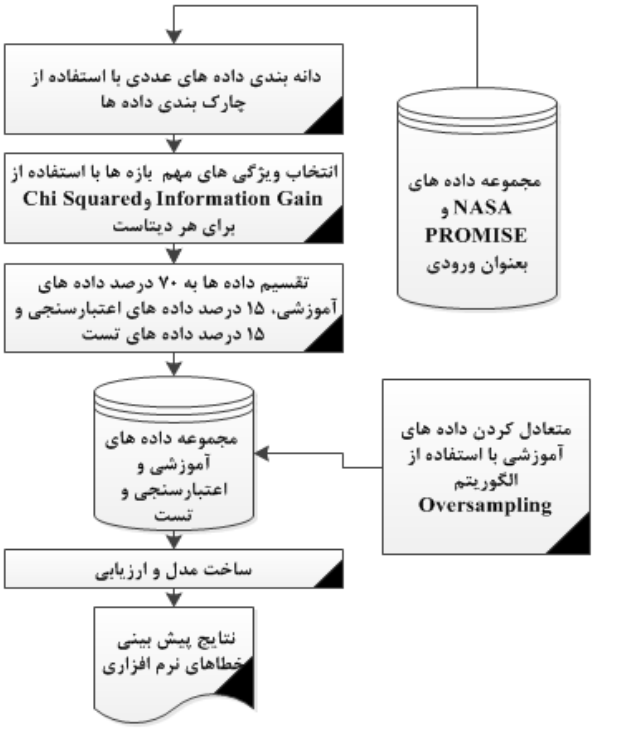
مقالات [۳۰-۳۱] یک بررسی کلی بر روی مجموعه داده‌ها و الگوریتم‌های مورد استفاده شده برای پیش‌بینی خطاهای نرم‌افزاری انجام داده اند. نتایج این مقالات نشان داده که تاکنون از شش دیتاست برای پیش بینی خطای نرم‌افزار استفاده شده و از دیتاست‌های PROMISE و NASA بیشتر استفاده شده است. با بررسی کارهای مرتبط در این زمینه به این نتیجه می‌رسیم که در دسته اول بیشتر الگوریتم‌ها سعی در به دست آوردن آهمیت ویژگی‌ها و استفاده آنها برای بهبود عملکرد سیستم بودند. کارهای ارایه شده در این دسته همچون [۴,۱۰,۱۲] نشان دادند که ویژگی‌های بکار رفته در هر دیتاست دارای وزن یکسانی نیستند و با متفاوت گرفتن وزن ویژگی‌ها می‌توان عملکرد سیستم را بهبود داد. لازم به ذکر است که در این روش‌ها فقط از یک دیتاست استفاده شده است و تعمیم‌پذیری الگوریتم‌ها بررسی نشده است. در دسته دوم بیشتر الگوریتم‌ها مثل [۲۲,۲۷] سعی در استفاده از کلمات به جای مقادیر عددی برای پیش‌بینی خطاهای نرم‌افزاری دارند. مهم‌ترین ایراد این روش‌ها نبود دیتاست بزرگ کلمات برای پیش‌بینی خطا و عدم تعمی‌پذیری الگوریتم‌های ارایه شده می‌باشد. در دسته سوم سعی در استفاده از دیتاست‌های عددی و تبدیل آنها به کلمات و استفاده از الگوریتم‌های یادگیری ماشین جهت بهبود عملکرد سیستم‌های ارایه شده می‌باشد.

۳- روش ارایه شده

وجود خطا در سیستم‌های نرم‌افزاری متداول است و در بیشتر پروژه‌ها مشهود است. معمولاً مهندسان نرم‌افزار در مرحله تست با تعریف نمونه‌های تست متفاوت، بیشتر خطاها را کشف می‌کنند. اما یافتن تمامی خطاها در مرحله تست به علت محدودیت زمانی و مالی برای بیشتر پروژه‌ها امکان پذیر نیست. به همین دلیل وجود یک سیستم که بتواند خطاهای نرم‌افزاری را پیش‌بینی کند، ضروری است. از مهم‌ترین چالش‌های اساسی سیستم‌های پیش‌بینی خطاهای نرم‌افزاری مقادیر عددی ویژگی‌ها، عدم تعمیم پذیری و نامتعادل بودن نسبت داده‌های خطا به داده‌های غیر خطا می‌باشند. در این بخش یک روش مبتنی بر دانه‌بندی اطلاعات[36,37] برای پیش‌بینی خطاهای نرم‌افزاری ارایه می‌شود. نوآوری این مقاله تبدیل مقادیر عددی ویژگی‌های دیتاست‌ها به مقادیر بازه‌ای و محاسبه اهمیت هر یک از ویژگی‌ها با استفاده از مقادیر بازه‌ای، رفع عدم تعادل داده‌ها با استفاده از الگوریتم oversampling و تعمیم پذیری روش ارایه شده می-باشد. لازم به ذکر است که تبدیل داده‌های عددی به بازه‌ها جهت کاهش مقادیر ممکن و محاسبه اهمیت ویژگی‌ها برای هر دیتاستی که مقادیر آنها بصورت عددی می‌باشد، قابل انجام است. منظور از تعمیم‌پذیری این است که بتوان با داده‌های یک یا چند پروژه یک دیتاست بعنوان داده‌های آموزشی، یادگیری مدل را انجام داد و با داده‌های هر یک از پروژه‌های دیگر آن دیتاست مدل ارایه شده را ارزیابی کرد. همچنین برای دیتاست‌های متفاوت، ابتدا ویژگی‌های مشترک آنها بدست آمده و با یکی از دیتاست‌ها یادگیری مدل انجام می‌شود و با دیتاست دیگر مدل ارایه شده مورد ارزیابی قرار می‌گیرد.

این معماری از چهار مرحله تشکیل شده است. در مرحله اول داده‌های عددی که نشان دهنده ویژگی‌های پروژهای مختلف دیتاست PROMISE و NASA می‌باشد توسط الگوریتم ارایه شده به بازه تبدیل می‌شوند. در مرحله دوم بر اساس ترکیبی از روش‌های Information Gain و Chi Squared پنج ویژگی مهم هر مجموعه داده انتخاب می‌گردد. لازم به ذکر است که این پنج ویژگی برای هر مجموعه داده و هر پروژه می‌تواند متفاوت باشد. در مرحله سوم، مساله عدم تعادل داده‌ها حل می‌شود. لازم به ذکر است که مسئله خطا در سیستم‌های نرم افزاری یک مسئله غیرمتعادل است که میزان نسبت ماژول‌های خطا به ماژول‌های غیرخطا خیلی کم است. برای حل این مساله از روش افزودن داده تصادفی به داده‌های آموزشی استفاده می‌شود. پس از انتخاب ویژگی‌ها شش الگوریتم مهم طبقه‌بندی برای پیش‌بینی خطاهای نرم‌افزاری در روش ارایه شده به کار گرفته می‌شود و نتایج حاصله با کارهای مرتبط مقایسه می‌شود. در ادامه هر یک از این مراحل توضیح داده می‌شود. معماری ارایه شده در این

مقاله در شکل۱ نشان داده شده است.



شکل (۱): مدل ارایه شده

۳-۱. رفع عدم تعادل داده‌ها و الگوریتم دانه بندی

دانه بندی داده‌های عددی متضمن آن است که بتوان بازه مناسب جهت تبدیل مقادیر عددی به کلمات LOW، MEDIUM و HIGH پیدا کرد. دلیل انتخاب این سه کلمه این است که در مقالات دیگر، همچون [۵] نیز از این سه کلمه استفاده شده است.

برای مدل‌سازی این کلمات الگوریتمی معرفی می‌شود. لازم به ذکر است که در [۵] از میانگین و ماکزیمم و مینیمم جهت تبدیل مقادیر عددی به بازه‌ها استفاده شده است. روشی که در [۵] ارائه شده است به این صورت است که مقادیر عددی هر ویژگی به یکی از کلمات LOW ، MEDIUM و HIGH تبدیل می شود. اگر مقدار هر ویژگی بیشتر از (Max+Avg)/2 باشد، مقدارآن ویژگی به HIGH تبدیل می شود. اگر مقدار هر یک از ویژگی ها بین (Max+Avg)/2 - Avg/4 و (Max+Avg)/2+ Avg/4 باشد، مقدار آن ویژگی با MEDIUM جایگزین می شود، در غیر این صورت مقدار آن ویژگی به LOW تغییر می کند. این الگوریتم نمی تواند محدوده بهینه برای تبدیل داده-های عددی به هر یک از سه کلمه را پیدا کند. به همین دلیل دقت روش [۵] پایین است. همچنین روش ارایه شده در [۵] نسبت به داده‌های پرت حساس است و با افزودن داده‌های پرت میانگین داده‌ها تغییر یافته و بازهای تبدیل اعداد به کلمات تغییر می‌یابد.به همین دلیل در این مقاله از روش مبتنی بر چارک بندی داده‌ها استفاده شده است. چارک بندی داده‌ها در مقابل داده پرت مقاوم هستند و باعث تغییر بازه های تبدیل اعداد به کلمات نمی‌شوند. در الگوریتم ارایه شده ورودی، مقادیر ویژگی‌های دیتاست‌های PROMISE و NASA می‌باشد.

این الگوریتم در شکل ۲ نمایش داده شده است. روند کار الگوریتم به این صورت است که ابتدا چارک اول و چارک سوم داده‌های ورودی برای هر ویژگی محاسبه می‌شود. برای محاسبه چارک اول و سوم، هر ویژگی را ابتدا به صورت صعودی مرتب می‌کنیم و بر اساس داده های مرتب شده چارک اول و سوم هر ویژگی را به دست می‌آوریم. داده‌هایی که مقدار آنها کمتر از چارک اول باشد بعنوان LOW در نظر گرفته می‌شود. داده هایی که مقادیر آنها بیشتر از چارک سوم باشد بعنوان HIGH در نظر گرفته می‌شود و بقیه داده ها بعنوان MEDIUM در نظر گرفته می‌شود. خروجی این الگوریتم برای مقدار هر ویژگی شامل یکی از مقادیر LOW ، MEDIUM و HIGH می‌باشد.

```
Information Granulation Algorithm ()
Input: PROMISE and NASA Datasets
Output: Linguistic Dataset
1. Q1= (First Quartile) and Q3= (Third Quartile)
2.  for i=1 to Count(features) do:
3.      Sort(feature[i][1..number of samples]
4.      Q1= feature[i][int((number of samples +1)/4)]
5.      Q3= feature[i][int(3*(number of samples +1)/4)]
6.      for j=1 to number of samples do:
7.          if feature[i][j] < Q1 then:
8.              feature[i][j] = LOW;
9.          else if feature[i][j] > Q3 then:
10.             feature[i][j] = HIGH;
11.         else:
12.             feature[i][j] = MEDIUM;
13.         end if
14.     end for
13. end for
```

شکل (۲): الگوریتم دانه بندی داده های عددی

۳-۲. الگوریتم انتخاب ویژگی‌های مهم

پس از دانه‌بندی داده‌های عددی باید بتوانیم ویژگی‌های مهم را برای هر مجموعه داده با استناد به بازه‌ها بدست بیاوریم. برای به دست آوردن اهمیت ویژگی‌ها از دو پارامتر مهم استفاده می‌شود. ابتدا Information Gain ویژگی‌ها را با استفاده از رابطه زیر محاسبه می نماییم.

$$I\,G(C,A) = H(C) - H(C|A) \qquad (۱)$$

در این رابطه C متغیر برچسب‌دار می‌باشد که نشان دهنده خطا یا عدم خطا می‌باشد. متغیر A نشان دهنده مقدار ویژگی می‌باشد. و تابع H آنتروپی را نشان می‌دهد [۳۸].

این رابطه را برای تک تک ویژگی‌ها محاسبه می‌کنیم. هر ویژگی که مقدار بالاتر Information Gain را داشت، اهمیت بالاتری دارد. سپس chi Squared با اعمال رابطه ۲ محاسبه می‌شود.

$$CHI(a,c) = \frac{(N \times (XZ - YW))}{((X+W) \times (Y+Z) \times (X+Y) \times (W+Z))} \quad (۲)$$

در این رابطه X نشان دهنده تعداد دفعاتی است که مقدار ویژگی a با برچسب کلاس c باهم اتفاق می‌افتند. Y نشان دهنده تعداد دفعاتی که مقدار ویژگی a بدون برچسب کلاس c اتفاق می‌افتند. W نشان دهنده تعداد دفعاتی که برچسب کلاس c بدون مقدار ویژگی a اتفاق می‌افتند و Z تعداد دفعاتی که نه ویژگی a و نه کلاس c اتفاق می افتد. N نیز مبین تعداد داده ها می‌باشد. مقدار بالاتر این پارامتر برای هر ویژگی نشان دهنده اهمیت بالای آن ویژگی است. در نهایت پس از محاسبه این دو پارامتر می توان با استفاده از رابطه ۳ اهمیت هر ویژگی را به دست آورد.

$$|v_a| = \sqrt{(IG_a)^2 + (CHI_a)^2} \quad (۳)$$

پس از محاسبه اهمیت ویژگی‌ها و جدا سازی داده‌های آموزشی و تست، از پنج ویژگی مهم‌ترکه دارای امتیاز بالاتری بودند، جهت طبقه‌بندی و ارزیابی روش ارایه شده استفاده می‌شود. برای ارزیابی ابتدا همه ویژگی‌ها در نظر گرفته شده و نتایج آنها بدست آمد. سپس با کم کردن تعداد ویژگی‌ها و مقایسه نتایج، بهترین نتایج با پنج ویژگی حاصل شد. برای کم کردن ویژگی‌ها ابتدا ویژگی-هایی که دارای امتیازات خیلی پایین بودند از داده‌ها حذف شده و مدل ساخته می‌شود. همین عمل تا رسیدن به بهترین نتیجه ادامه پیدا می‌کند. لازم به ذکر است که در ادامه از شش روش طبقه‌بندی استفاده کردیم. بر اساس نتایج، بهترین عملکرد بین شش الگوریتم با روش رگرسیون لاجستیک (LR) بدست آمده است.

۳-۳. رفع عدم تعادل داده‌ها

در اکثر پروژه‌های نرم‌افزاری نسبت ماژول‌های بدون خطا به ماژول‌های دارای خطا عدد بالایی است و این مسئله باعث عدم تعادل در مسئله پیش‌بینی می‌شود. برای حل مسئله عدم تعادل یکی از دو روش undersampling یا oversampling انجام می‌شود. در روش undersampling از داده‌های بدون خطا کم می‌کنند تا نسبت داده‌های بدون خطا به داده‌های خطا کمتر شود. در حالی که در روش oversampling به داده‌های خطا اضافه می‌کنند تا نسبت ماژول‌های بدون خطا به ماژول‌های دارای خطا کمتر گردد. در این مقاله از روش oversampling استفاده می‌شود تا نسبت داده‌ها متعادل‌تر می‌شود. منظور از متعادل‌سازی داده‌ها این است که میزان درصد خطاها برای هر پروژه در هر دیتاست نزدیک و مساوی با میزان درصد غیر خطا بودن آن پروژه در آن دیتاست باشد.

۴- ارزیابی

جهت ارزیابی راهکار پیشنهادی، از دو دیتاست PROMISE وNASA استفاده می‌نماییم[۳۴,۳۵]. دیتاست PROMISE دارای ۳۲ پروژه مختلف است که هر کدام از آنها دارای چندین نسخه هستند [۳۴]. دلیل انتخاب دیتاست PROMISE این است که اولا این مجموعه داده از داده‌های پلتفرم مشترک مشتق شده‌اند و به‌طور عمومی در دسترس هستند و به عنوان مجموعه داده‌های پایه برای مطالعات پیش‌بینی خطاهای نرم‌افزاری در نظر گرفته می‌شوند. دوما این مجموعه داده‌ها متن باز هستند و دارای ویژگی‌های عمومی هستند که برای استفاده مستقیم از آنها و ارزیابی عملکرد مدل‌ها مفید است. محققان می‌توانند از آنها برای ارزیابی، مقایسه و تکرار مطالعات خود استفاده کنند. سوما این مجموعه داده‌ها نامتعادل هستند، و به ما امکان می دهد روش پیشنهادی خود را برای رسیدگی به مشکل عدم تعادل کلاس اعمال و ارزیابی کنیم.

از بین پروژه های دیتاست PROMISE ، ۹ پروژه جهت ارزیابی‌های آتی انتخاب شدند. انتخاب ۹ پروژه از بین پروژه‌ها به دو دلیل بوده است. ابتدا برخی مقالات مانند [۵] نیز از این ۹ پروژه استفاده کرده اند. ثانیا عدم تعادل در این پروژه ها بیشتر بود. میزان درصد خطا برای این ۹ پروژه در جدول ۱ نشان داده شده است.

جدول ۱: میزان خطای نه پروژه از دیتاست PROMISE

ردیف	نام پروژه	درصد خطا بر حسب (٪)
۱	ant	۲۰.۶۹
۲	camel	۲۰.۱۹
۳	log4j	۵۷.۹۱
۴	jedit	۱۷.۳۲
۵	synapse	۲۵.۵۱
۶	velocity	۵۷.۴۳
۷	xalan	۵۴.۴۰
۸	xerces	۳۹.۸۱
۹	prop	۱۳.۲۲

دیتاست NASA دارای ۲۱ ویژگی می‌باشد. این دیتاست شامل پروژه‌های مختلفی می‌باشد که در این مقاله از پنج پروژه برای ارزیابی استفاده می‌شود. به دلیل عدم تعادل داده‌ها و همچنین میزان استفاده این پروژه ها در مقالات، این پروژه‌ها انتخاب شدند. لازم به ذکر است پنج پروژه ای که از این دیتاست انتخاب شده به همراه میزان درصد خطا برای آنها در جدول۲ نشان داده شده است.

جدول ۲: میزان خطای پنج پروژه از دیتاست NASA

ردیف	نام پروژه	درصد خطا بر حسب (٪)
۱	PC1	۱۲.۸
۲	CM1	۸.۱
۳	JM1	۲۰.۸
۴	KC1	۲۵.۳
۵	KC2	۱.۸

لازم به ذکر است که عدم تعادل در هر دو دیتاست در جداول ۱ و ۲ نشان داده شده است. هر پروژه که میزان خطای آن نزدیک به ۵۰ درصد باشد متعادل است که با توجه به جداول فوق در اکثر پروژه‌ها این تعادل وجود ندارد. منظور از درصد خطا که در جدول ۱ و ۲ نشان داده شده است این است که چه تعداد خطا در هر کدام از پروژه‌های هر مجموعه داده وجود دارد. ما از دو دیتاست PROMISE و NASA برای ارزیابی روش ارایه شده در این مقاله استفاده کردیم. هر کدام از این دیتاست‌ها شامل پروژه‌های مختلفی هستند. که ما برای هر دیتاست، این پروژه‌ها را به دلیل میزان استفاده در سایر مقالات و داشتن نسبت عدم تعادل بین تعداد نمونه‌های خطا به نمونه‌های غیر خطا انتخاب کردیم. هر مجموعه داده حاوی چندین ویژگی است که آخرین ویژگی هر مجموعه داده خطا دار بودن یا عدم خطادار بودن را نشان می‌دهد. با شمارش تعداد نمونه‌ها با برچسب‌های خطا و تقسیم آن بر تعداد کل داده‌های هر پروژه از مجموعه داده این اعداد به دست آمده است. بعنوان مثال در پروژه ant از مجموعه داده PROMISE ما تعداد نمونه‌هایی که مقدار آخرین ویژگی آن بزرگتر از صفر بوده است را شمردیم که تعداد نمونه‌های خطا در این پروژه برابر ۳۵۰ می‌باشد. عدد بدست آمده را بر تعداد کل نمونه پروژه ant که برابر ۱۶۹۲

می‌باشد، تقسیم کرده و عدد ۲۰.۶۹ درصد حاصل شده است که نشان دهنده میزان خطای همین پروژه می‌باشد. هدف از این دو جدول این است که نشان دهیم هر مجموعه داده که استفاده می‌شود عدم تعادل در آن وجود دارد و باید قبل از استفاده از الگوریتم‌های یادگیری ماشین مساله عدم تعادل باید برطرف شود. بعنوان مثال در جدول ۲ میزان خطا برای پروژه CM1 در دیتاست NASA برابر ۸ درصد می‌باشد و ۹۲ درصد این پروژه برچسب داده‌ای بدون خطا دارد در صورت عدم استفاده از الگوریتم‌های رفع تعادل، الگوریتم‌های طبقه‌بندی نتایج مطلوبی ندارند. هدف ما طبقه‌بندی داده‌ها بر اساس دو کلاس خطا و عدم خطا می‌باشد. دلیل استفاده از الگوریتم متعادل سازی داده‌ها ایجاد توزیع مناسب بین نسبت نمونه‌های خطا به نمونه‌های غیز خطا در هر یک از پروژه های هر مجموعه داده می‌باشد. به همین دلیل هدف از رفع عدم تعادل این است که میزان خطای هر پروژه را با استفاده از تکنیک oversampling به ۵۰ درصد برسانیم. در شکل ۳ نمونه‌ای از پروژه Ant از دیتاست PROMISE مشاهده می‌شود. در این شکل هفت نمونه از این پروژه مشاهده می‌شود. خط اول این شکل نشان دهنده نام ویژگی‌ها و خط های بعدی مقادیر را نشان می‌دهد. با توجه به شکل ۳ مشاهده می‌شود که نمونه سوم برچسب خطا دارد در حالی که بقیه نمونه‌ها عدم خطا دار را نشان می‌دهد.

```
name,wmc,dit,noc,cbo,rfc,lcom,ca,ce,npm,lcom3,loc,dam,moa,mfa,cam,ic,cbm,amc,max_cc,avg_cc,bug
ant,11,4,2,14,42,29,2,12,5,0.725,1,1,2,0.9,3,4,34,3,1.2727,0
ant,14,1,1,8,32,49,4,4,12,0.7,257,1,0,0,0.,0,0,16.,6,1.6429,1
ant,3,2,0,1,9,0,0,1,1,0,58,1,1,0.714285714,0.,1,1,17,1,0.6667,0
ant,12,3,0,12,37,32,0,12,12,0.8,310,1,1,0.78,0.5,0,0,24,3,1.4167,0
ant,6,3,0,4,21,1,0,4,6,0.7,136,1,0,0.880952381,0.416666667,2,2,21,1,0.8333,0
ant5,1,0,3,15,0,2,1,4,0.5,137,1,0,0,0.9,0,0,25.2,4,1.8,0
ant4,4,0,6,32,6,0,6,1,2,325,0,0,0.963855422,0.75,3,5,80.25,11,5.25,0
```

شکل ۳: چند نمونه از داده‌های پروژه ant از دیتاست PROMISE

جدول ۳: ویژگی‌های پروژه‌های دیتاست PROMISE

ویژگی‌های PROMISE			
توصیف ویژگی	**ویژگی**	توصیف ویژگی	**ویژگی**
Lack of Cohesion in Methods	lcom3	Weighted Methods for Class	wmc
Lines of Code	loc	Depth of Inheritance Tree	dit
Data Access Metric	dam	Number of Children	noc
Measure of Aggregation	moa	Coupling between Objects	cbo
The measure of Functional Abstraction	mfa	Response for Classes	rfc
Cohesion among Methods of Class	cam	Lack of Cohesion of Methods	lcom
Inheritance Coupling	ic	Afferent Coupling	ca
Coupling between Methods	cbm	Efferent Coupling	ce
Average Method Complexity	amc	Number of Public Methods	npm
-	-	Cyclomatic complexity (mux)	max-cc

ویژگی‌های موجود برای هر یک از دیتاست‌ها در جدول‌های ۳و ۴ نشان داده شده است. این ویژگی‌ها و توصیف آنها در این جداول نشان داده شده است. از این ویژگی‌ها و مقادیر آنها برای ارزیابی الگوریتم ارایه شده در این مقاله استفاده می‌شود.

جدول ۴: ویژگی‌های پروژه‌های دیتاست NASA

ویژگی‌های NASA			
توصیف ویژگی	**ویژگی**	توصیف ویژگی	**ویژگی**
McCabe cyclomatic complexity	v(G)	Number of blank lines	LOCb
McCabe design complexity	iv(G)	Number of comment-only lines	LOCc
McCabe essential complexity	ev(G)	Number of code-only lines	LOCe
Halstead program length, N=N1+N2	N	Number of lines that contain both code and comment	LOCec
Halstead program volume,V=N*log 2(p1+p2)	V	Total number of lines	L.O.C
Halstead program difficulty ,D=(p1/2)*(N2/p 2)	D	Number of branches	B.R.
Halstead program level, L=1/D	L	Number of unique operators	P1
Halstead program effort, E=D*V	E	Total number of operator	N1
Halstead program time,T=E/18	T	Number of unique operands	P2
Halstead error estimate, $B=E^{2/3}/3000$	B	Total number of operands	N2
-	-	Halstead intelligent content,I= (1/D)*V	I

برای ارزیابی روش ارایه شده از پارامتر AUC و Accuracy و F-measure استفاده می‌نماییم. علاوه بر این نتایج حاصل را با شش روش یادگیری ماشین که در دیگر مقالات [۵] استفاده شده‌اند، مقایسه می‌کنیم. شش روش به ترتیب درخت تصمیم(DT)، جنگل تصادفی(RF)، پرسپترون چند لایه (MP)، روش بیزین (NB) و درخت تصادفی (RT) و رگرسیون لاجستیک(LR) می‌باشند.

۴-۱. پارامترهای ارزیابی

در این قسمت پارامترهای ارزیابی معرفی می‌شود. برای ارزیابی و مقایسه روش ارایه شده در این مقاله با سایر روش‌ها از سه پارامتر Accuracy و F-measure و AUC استفاده می‌شود.

۴-۱-۱. Accuracy

این پارامتر برای ارزیابی درستی[2] استفاده می شود. برای به دست اوردن مقدار این پارامتر، تعداد تشخیص‌های درست به تعداد کل تشخیص‌ها تقسیم می‌شود که در رابطه ۴ نشان داده شده است.

$$Accuracy = \frac{True\ Posistive + True\ Negative}{Total\ Number\ of\ Data} \times 100 \quad (۴)$$

۴-۱-۲. F-measure

برای بدست آورن این پارامتر از میانگین دو پارامتر Recall و Precision استفاده می‌شود. برای محاسبه هر کدام از این پارامترها از روابط ۵ و ۶ و ۷ استفاده می‌شود.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \times 100 \quad (۵)$$

$$Precision = \frac{\text{True Positive}}{\text{True Positive+False Positive}} \times 100 \quad (۶)$$

$$F - measure = \frac{2 \times Recall \times Precision}{Reall + Precision} \times 100 \quad (۷)$$

۴-۱-۳. AUC

نمودار ROC[3] به صورت گرافیکی نمایش می‌دهد که چقدر یک مدل دسته‌بندی عملکرد خوبی دارد. AUC مساحت زیر نمودار، یک معیار عددی است که از نمودار ROC مشتق می‌شود و عملکرد کلی یک مدل را خلاصه می‌کند.

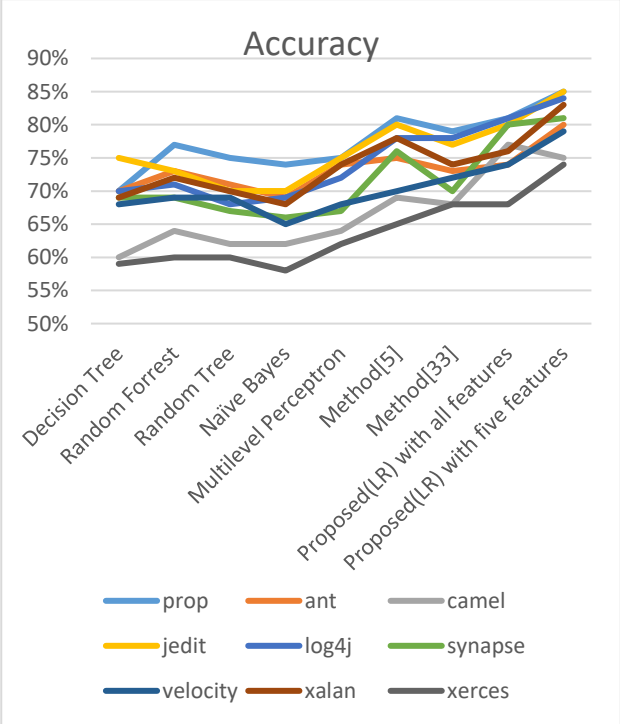
۴-۲. ارزیابی بر روی داده‌های PROMISE

برای ساخت مدل پیش بینی خطاهای نرم‌افزاری، برای هر یک از پروژه‌ها ۷۰ درصد داده بعنوان داده آموزشی، ۱۵ درصد بعنوان داده اعتبارسنجی و ۱۵ درصد باقیمانده بعنوان داده تست در نظر گرفته می‌شود. در این مقاله با داده-های آموزشی و اعتبارسنجی بر اساس الگوریتم‌های طبقه‌بندی مدل پیش‌بینی خطاهای نرم‌افزاری ساخته می‌شود. سپس با داده‌های تست روش ارایه شده ارزیابی می‌شود. به عنوان مثال برای پروژه Prob پس از مشخص شدن داده‌های آموزشی و اعتبارسنجی و تست، ابتدا مقادیر ویژگی‌های مجموعه داده PROMISE با استفاده از الگوریتم ارایه شده دانه‌بندی شده و پنج ویژگی مهم آن با استفاده از الگوریتم انتخاب ویژگی‌های مهم مطرح شده در بخش ۳-۲، محاسبه شد. برای ارزیابی ابتدا همه ویژگی‌ها با بازه‌ها در نظر گرفته شد و نتایج آنها بدست آمد. سپس با کم کردن تعداد ویژگی‌ها و مقایسه نتایج، بهترین نتایج با پنج ویژگی حاصل شد. برای کم کردن ویژگی‌ها ابتدا ویژگی با کمترین امتیاز حذف شد و مدل یادگیری بدون آن ویژگی ساخته شد و مورد ارزیابی قرار گرفت و این عمل تا زمان رسیدن به بهترین نتیجه ادامه پیدا کرد. در نهایت پارامترهای ارزیابی Accuracy ، F-measure و AUC محاسبه شد. در جدول ۵ ویژگی‌های انتخاب شده برای هر یک از نه پروژه دیتاست PROMISE نشان داده شده است.

جدول ۵:پنج ویژگی با بالاترین امتیاز انتخاب شده در دیتاست PROMISE

نام پروژه	ویژگی‌های انتخاب شده				
prop	wmc	rfc	loc	ca	moa
ant	noc	lcom	ca	npm	dam
camel	wmc	lcom	ce	moa	cam
jedit	wmc	dit	lcom	rfc	loc
Log4j	dit	rfc	ca	dam	moa
synapse	dit	noc	lcom	dam	mfa
velocity	wmc	cbo	rfc	ca	amc
xalan	wmc	dit	noc	moa	lcom
xerces	dit	rfc	lcom	npm	loc

در شکل ۴ میانگین دقت نتایج کار این مقاله و مقایسه آن با روش‌های قبلی نمایش داده شده است. لازم به ذکر است که برای ارزیابی، روش ارایه شده در این مقاله با روش‌های ارایه شده در [۵] و [۳۳] مقایسه شده است. علاوه بر این برای انجام ارزیابی از شش روش طبقه‌بندی استفاده شده است. که روش مبتنی بر LR بر اساس نتایج، بهترین عملکرد را دارد.



شکل(۴): مقایسه دقت بدست آمده روش ارایه شده با سایر روش‌ها در دیتاست PROMISE

با توجه به شکل ۴ مشاهده می‌شود که دقت بدست آمده برای نه پروژه دیتاست PROMISE با روش ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است. بعنوان مثال در پروژه jedit پارامتر Accuracy بدست آمده با روش ارایه شده LR برای پنج ویژگی با امتیاز بالا در این مقاله برابر ۸۵٪ است ولی با روش جنگل تصادفی (۷۳٪)، درخت تصمیم (۷۵٪)، روش بیزین (۷۰٪)، درخت تصادفی (۷۰٪) ، پرسپترون چند لایه (۷۵٪) و روش ارایه شده در [۵] برابر ۸۰ ٪ و روش ارایه شده در [۳۳] برابر ۷۷٪ و با روش LR بدون انتخاب ویژگی‌ها برابر ۸۰ ٪ است. همانطور که مشاهده می‌شود روش ارایه شده در این مقاله ۵٪ عملکرد بهتری دارد. به عبارت دیگر انتخاب پنج ویژگی برتر نسبت به حالتی که تمامی ویژگی‌ها را در نظر بگیریم، نتایج بهتری دارد.

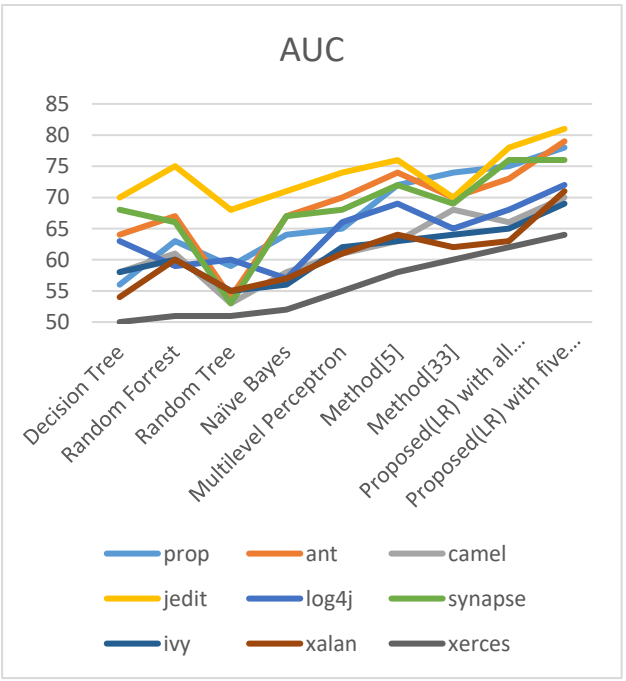
در شکل ۵ نتایج AUC نشان داده شده است. نتایج نشان می‌دهد که روش ارایه شده برای نه پروژه نسبت به سایر روش‌ها عملکرد بهتری دارد. بعنوان مثال در

[2] Accuracy

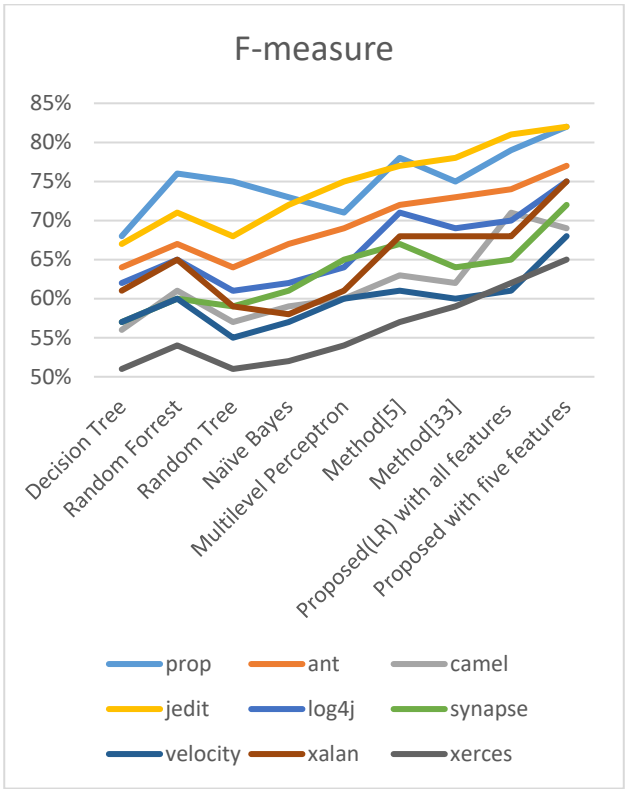
[3] Receiver Operating Characteristic

پروژه Prob پارامتر AUC بدست آمده با روش ارایه شده و بکارگیری روش LR برای پنج ویژگی انتخاب شده در این مقاله برابر ۷۸٪ است ولی با روش جنگل تصادفی (۶۳٪)، درخت تصمیم (۵۶٪)، روش بیزین (۶۴٪)، درخت تصادفی (۵۹٪) ، پرسپترون چند لایه (۶۵٪) و روش ارایه شده در [۵] برابر ۷۲ ٪ و روش ارایه شده در [۳۳] برابر ۷۴ ٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر۷۵ ٪ است که روش ارایه شده در این مقاله ۳٪ عملکرد بهتری دارد. به عبارت دیگر انتخاب پنج ویژگی نسبت به حالتی که تمامی ویژگی‌ها را در نظر بگیریم، نتایج بهتری دارد.

در شکل ۶ نتایج F-measure نشان داده شده است. نتایج نشان می‌دهد که روش ارایه شده برای نه پروژه نسبت به سایر روش‌ها عملکرد بهتری دارد. بعنوان مثال در پروژه ant پارامتر F-measure بدست آمده با روش ارایه شده و بکارگیری روش LR و با در نظر گرفتن پنج ویژگی مهم در این مقاله برابر ۷۷٪ است و با روش جنگل تصادفی (۶۷٪)، درخت تصمیم (۶۴٪)، روش بیزین (۶۷٪)، درخت تصادفی (۶۴٪) ، پرسپترون چند لایه (۶۹٪) و روش ارایه شده در [۵] برابر ۷۲ ٪ و روش ارایه شده در [۳۳] برابر ۷۳ ٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر ۷۴ ٪ است که روش ارایه شده در این مقاله ۳٪ عملکرد بهتری دارد. به عبارت دیگر انتخاب پنج ویژگی نسبت به حالتی که تمامی ویژگی‌ها را در نظر بگیریم، نتایج بهتری دارد. لازم به ذکر است که در این حالت برای هر پروژه از هر دیتاست ، داده آموزشی و اعتبارسنجی و تست از همان پروژه در همان دیتاست انتخاب شده است. به عبارت دیگر ساخت مدل یادگیری و ارزیابی مدل ساخته شده برای هر پروژه بصورت مستقل انجام پذیرفته است.



شکل (۵): مقایسه AUC روش ارایه شده با سایر روش‌ها در دیتاست PROMISE



شکل (۶): مقایسه F-measure روش ارایه شده با سایر روش‌ها در دیتاست PROMISE

۴-۳. ارزیابی بر روی داده‌های NASA

برای ساخت مدل پیش بینی خطاهای نرم‌افزاری برای هر یک از پروژه‌های این دیتاست، ۷۰ درصد داده بعنوان داده آموزشی، ۱۵ درصد بعنوان داده اعتبارسنجی و ۱۵ درصد باقیمانده بعنوان داده تست در نظر گرفته می‌شود. به عنوان مثال برای CM1 پس از مشخص شدن داده‌های آموزشی و اعتبار سنجی و تست، ابتدا مقادیر ویژگی‌های این دیتاست بر اساس الگوریتم ارایه شده در این مقاله دانه‌بندی می‌شود و سپس پنج ویژگی مهم بر اساس روابط ۱ تا ۳ ارایه شده این مقاله محاسبه می‌شود. در مرحله آخر بر اساس روش oversampleing مساله عدم تعادل ماژول‌های خطا و عدم خطا برای داده-های آموزشی انجام می‌شود. در نتیجه AUC، F-measure و Accuracy محاسبه می‌شود. در جدول ۶ پنج ویژگی با امتیاز بالا که با روابط ۱ تا ۳ محاسبه شده، نشان داده شده است.

جدول۶:پنج ویژگی با امتیاز بالا انتخاب شده در دیتاست NASA

نام پروژه	ویژگی‌های انتخاب شده				
PC1	LOCe	B.R	Iv(G)	Ev(G)	N
CM1	LOCb	LOCe	loc	P1	V
JM1	LOCe	LOCec	N1	P2	D
KC1	LOCc	B.R	D	L	B
KC2	P1	V(G)	L	N	I

در جدول ۷ پارامتر Accuracy و مقایسه آن برای پنج پروژه دیتاست NASA با روش‌های قبلی به دست آمده است.

جدول ۷: مقایسه دقت برای داده های NASA

نام پروژه	PC1	CM1	JM1	KC1	KC2
Accuracy(%)					
RF	۶۳	۶۲	۵۷	۶۶	۶۱
NB	۶۷	۶۴	۶۵	۶۴	۶۴
DT	۵۹	۵۸	۵۷	۵۹	۶۱
RT	۷۲	۷۰	۶۶	۷۴	۶۱
MP	۷۵	۷۴	۶۳	۷۰	۷۱
[۵]	۷۳	۷۳	۶۴	۷۰	**۷۵**
[۳۳]	۷۱	۷۵	۶۲	۶۷	۷۰
Proposed LR with all features	۷۷	۷۸	۶۴	**۷۵**	۷۱
Proposed LR with five features	**۸۳**	**۸۱**	**۷۲**	۷۱	**۷۵**

با توجه به جدول ۷ مشاهده می‌شود که دقت ارایه شده برای پنج پروژه دیتاست NASA در روش ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است. بعنوان مثال در پروژه CM1 پارامتر Accuracy بدست آمده برای روش ارایه شده در این مقاله با روش LR و با در نظر گرفتن پنج ویژگی مهم برابر ۸۱٪ است که نسبت به روش جنگل تصادفی (۶۲٪)، درخت تصمیم (۵۸٪)، روش بیزین (۶۴٪)، درخت تصادفی (۷۰٪) ، پرسپترون چند لایه (۷۴٪) و روش ارایه شده در [۵] برابر ۷۳ ٪ و روش ارایه شده در [۳۳] برابر ۷۵ ٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر ۷۸ ٪ است که روش ما ۳٪ عملکرد بهتری دارد. این نتایج نشان می‌دهد که استفاده از روش مبتنی بر انتخاب ویژگی‌ها عملکرد بهتری دارد. در جدول ۸ پارامتر F-measure و مقایسه آن برای پنج پروژه دیتاست NASA با روش‌های قبلی به دست آمده است

.

جدول۸ : مقایسه F-measure برای داده های NASA

نام پروژه	PC1	CM1	JM1	KC1	KC2
F-measure(%)					
RF	۶۸	۶۶	۶۸	۶۵	۶۶
NB	۶۳	۶۵	۶۴	۶۲	۵۹
DT	۵۸	۶۰	۶۲	۶۰	۵۸
RT	۶۸	۷۲	۶۹	۶۶	۶۵
MP	۶۹	۷۴	۶۸	۶۴	۶۶
[۵]	۷۲	۷۷	۷۲	۷۰	۷۰
[۳۳]	۷۴	۷۰	۷۱	۶۵	۶۸
Proposed LR with all features	۷۶	۷۴	۷۲	**۷۹**	۷۱
Proposed LR with five features	**۸۰**	**۸۲**	**۷۵**	۷۴	**۷۴**

با توجه به جدول۸ مشاهده می‌شود که F-measure برای پنج پروژه دیتاست NASA با روش ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است. بعنوان مثال در پروژه KC1 پارامتر F-measure بدست آمده برای روش ارایه شده در این مقاله برابر ۷۴٪ است که نسبت به روش جنگل تصادفی (۶۵٪)، درخت تصمیم (۶۰٪)، روش بیزین (۶۲٪)، درخت تصادفی (۶۶٪) ، پرسپترون چند لایه (۶۴٪) و روش ارایه شده در [۵] برابر ۷۰ ٪ و روش ارایه شده در [۳۳] برابر ۶۵ ٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر ۷۹ ٪ است. لازم به ذکر است که در حالت کلی و با در نظر گرفتن تمامی پروژه‌های این دیتاست روش LR مبتنی بر انتخاب پنج ویژگی مهم نتایج بهتری در مقایسه با سایر روش‌ها دارد. در جدول۹ پارامتر AUC و مقایسه آن برای پنج پروژه دیتاست NASA با روش‌های قبلی به دست آمده است.

جدول۹: مقایسه AUC برای داده های NASA

نام پروژه	PC1	CM1	JM1	KC1	KC2
AUC(%)					
RF	۶۸	۶۴	۶۶	۷۱	۷۵
NB	۶۴	۶۸	۶۵	۷۱	۶۷
DT	۶۸	۵۹	۶۱	۶۷	۶۱
RT	۶۹	۷۳	۶۸	۷۳	۷۱
MP	۸۲	۷۷	۷۱	۷۹	۷۶
[۵]	۸۰	۷۸	۷۳	۷۵	۸۱
[۳۳]	۷۸	۷۵	۷۰	۷۰	**۸۵**
Proposed LR with all features	۸۱	۷۶	۷۱	**۸۷**	۷۸
Proposed LR with five features	**۸۵**	**۸۰**	**۷۷**	۸۰	۸۳

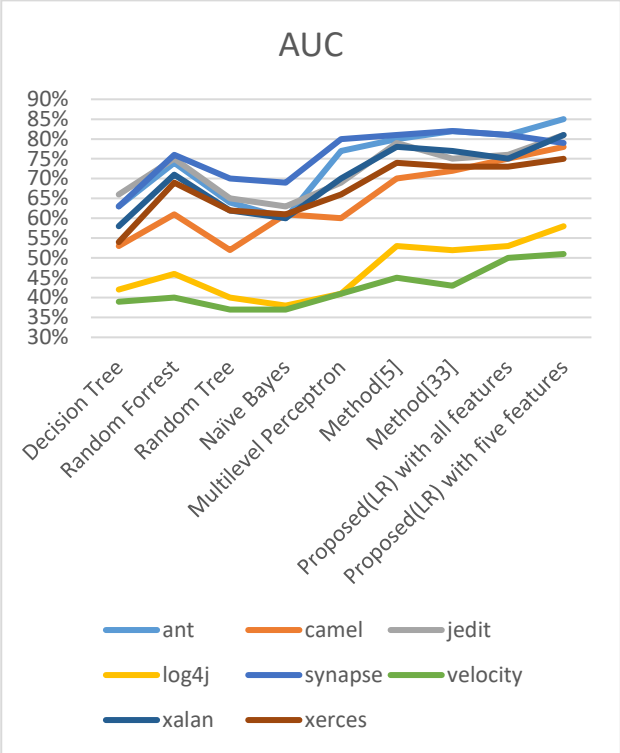
با توجه به جدول ۹ مشاهده می‌شود که AUC ارایه شده برای چهار پروژه دیتاست NASA در روش ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است. بعنوان مثال در پروژه JM1 پارامتر AUC بدست آمده برای روش ارایه شده در این مقاله برابر ۷۷٪ است که نسبت به روش جنگل تصادفی (۶۶٪)، درخت تصمیم (۶۱٪)، روش بیزین (۶۵٪)، درخت تصادفی (۶۸٪) ، پرسپترون چند لایه (۷۱٪) و روش ارایه شده در [۵] برابر ۷۳ ٪ و روش ارایه شده در [۳۳] برابر ۷۰ ٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر ۷۱ ٪ است که روش ما ۴٪ عملکرد بهتری دارد. لازم به ذکر است که در حالت کلی و با در نظر گرفتن تمامی پروژه‌های این دیتاست روش LR مبتنی بر انتخاب پنج ویژگی مهم نتایج بهتری در مقایسه با سایر روش‌ها دارد.

۴-۴. تعمیم پذیر الگوریتم ارایه شده

در این بخش هدف این است که بتوانیم تعمیم پذیری روش ارایه شده را نشان دهیم. دو نوع ارزیابی برای تعمیم‌پذیری در این مقاله انجام می‌شود. در ارزیابی نوع اول یکی از پروژه های هر دیتاست بعنوان داده آموزشی و اعتبارسنجی در نظر گرفته می‌شود و پروژه های دیگر همان دیتاست بعنوان داده تست در نظر گرفته می‌شود. این نوع ارزیابی برای بسیاری از شرکت‌ها کاربرد دارد. برای ساخت نسخه‌های بعدی از یک محصول می‌توان میزان خطای نسخه جدید را پیش‌بینی نمود تا سریعتر نسخه جدید تولید گردد. در ارزیابی نوع دوم داده های یک دیتاست بعنوان داده آموزشی و اعتبارسنجی در نظر گرفته شد و داده های دیتاست دیگر بعنوان تست در نظر گرفته شد. هدف از این ارزیابی این است که شرکت‌های نوپا و جدید که می‌خواهند محصول مشابه با محصول شرکت‌های دیگر تولید کنند، بتوانند میزان خطای محصول خود را بر اساس محصول شرکت‌های دیگر پیش‌بینی کنند و محصول خود را سریعتر روانه بازار کنند. هر یک از ارزیابی‌ها در ادامه بررسی می‌شود. در ارزیابی نوع اول یکی از پروژه‌های هر دیتاست را بعنوان داده آموزشی و اعتبارسنجی در نظر می‌گیریم و مدل یادگیری خود را بر اساس آن پروژه انجام می‌دهیم. سپس بقیه پروژه‌ها را بعنوان داده تست در نظر می‌گیریم. بدین منظور ازدیتاست PROMISE پروژه prob را بعنوان داده آموزشی و اعتبارسنجی در نظر می‌گیریم. همچنین از دیتاست NASA پروژه JM1 را بعنوان داده آموزشی و اعتبارسنجی در نظر می-گیریم. دلیل انتخاب این دو پروژه از دو دیتاست حجم داده زیاد آنها می‌باشد تا مدل یادگیری بتواند مدل کامل و جامعی ایجاد کند. ۸۵ درصد از هر پروژه مذکور بعنوان داده آموزشی و ۱۵ درصد باقیمانده، بعنوان داده اعتبارسنجی در

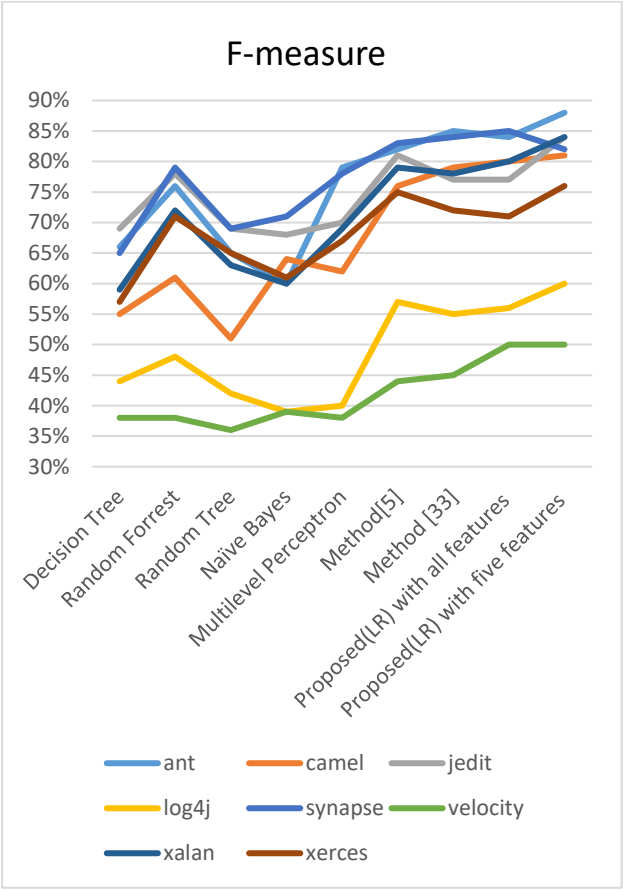
نظر گرفته می شود و مدل یادگیری برای هر کدام از دیتاست‌ها ساخته می‌شود. سپس هر یک از پروژه‌های باقیمانده در هر دیتاست بعنوان داده تست در نظر گرفته می‌شود. سپس پنج ویژگی مهم براساس روابط بیان شده در بخش قبلی محاسبه می‌شود.لازم به ذکر است ابتدا تمامی ویژگی‌ها در نظر گرفته شدند و مدل یادگیری و ارزیابی انجام شد. سپس با استفاده ار روابط (۱) و (۲)، ویژگی های با کمترین امتیاز حذف گشته و همین رویه تکرار گردید. پنج ویژگی انتخاب شده برای پروژه prob در مجموعه داده PROMISE و پنج ویژگی انتخاب شده برای پروژه JM1 بر اساس جدول ۵ برای ارزیابی بهترین نتایج را در پی داشتند. به دلیل حجم داده زیاد این دو پروژه در دو دیتاست، انتخاب پنج ویژگی این دو پروژه بهترین نتیجه را ایجاد کرد.

در شکل ۷ نتایج AUC نشان داده شده است. نتایج نشان می‌دهد که روش ارایه شده برای هشت پروژه نسبت به سایر روش‌ها عملکرد بهتری دارد. بعنوان مثال برای پروژه jedit پارامتر AUC بدست آمده با روش ارایه شده بهمراه روش LR برای پنج ویژگی انتخاب شده در این مقاله برابر ۸۱٪ است که نسبت به روش جنگل تصادفی (۷۵٪)، درخت تصمیم (۶۶٪)، روش بیزین (۶۳٪)، درخت تصادفی (۶۵٪) ، پرسپترون چند لایه (۶۹٪) و روش ارایه شده در [۵] برابر ۷۹ ٪ و روش ارایه شده در [۳۳] برابر ۷۵٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر۷۶ ٪ است که روش ارایه شده در این مقاله ۲٪ عملکرد بهتری دارد. به عبارت دیگر انتخاب پنج ویژگی نسبت به حالتی که تمامی ویژگی‌ها را در نظر بگیریم، نتایج بهتری دارد. لازم به یادآوری است که در این حالت یکی از پروژه‌ها معمولا پروژه با حجم داده زیاد بعنوان داده آموزشی و اعتبارسنجی در نظر گرفته می‌شود و داده‌های پروژه‌های دیگر همان دیتاست بعنوان داده تست لحاظ می‌گردد. به عبارت دیگر ساخت مدل یادگیری با یکی از پروژه‌های یک دیتاست انجام می‌شود و ارزیابی و تست مدل ساخته شده برای هر پروژه جداگانه انجام می‌شود. این حالت از ارزیابی را ارزیابی بین پروژه‌ای از یک دیتاست در نظر می‌گیریم. هدف از این ارزیابی این است که می توان با داده‌های یک نسخه از نرم‌افزار مدل یادگیری ساخت و تعداد خطاهای نسخه‌های آینده از همان نرم‌افزار را قبل از ساخت آن نسخه از نرم‌افزار را پیش‌بینی کرد.



شکل (۷): مقایسه AUC روش ارایه شده با سایر روش‌ها در دیتاست PROMISE

در شکل ۸ نتایج F-measure نشان داده شده است. نتایج نشان می‌دهد که روش ارایه شده برای هشت پروژه نسبت به سایر روش‌ها عملکرد بهتری دارد. لازم به ذکر است که پروژه prob بعنوان داده آموزشی بوده و مدل یادگیری بر اساس آن ساخته شده است. بعنوان مثال برای پروژه log4j پارامتر F-measure بدست آمده با روش ارایه شده با روش LR و با در نظر گرفتن پنج ویژگی مهم در این مقاله برابر ۶۰٪ است که نسبت به روش جنگل تصادفی (۴۸٪)، درخت تصمیم (۴۴٪)، روش بیزین (۳۹٪)، درخت تصادفی (۴۲٪) ، پرسپترون چند لایه (۴۰٪) و روش ارایه شده در [۵] برابر ۵۷ ٪ و روش ارایه شده در [۳۳] برابر ۵۵٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر ۵۶ ٪ است که روش ارایه شده در این مقاله ۳٪ عملکرد بهتری دارد. به عبارت دیگر انتخاب پنج ویژگی نسبت به حالتی که تمامی ویژگی‌ها را در نظر بگیریم، نتایج بهتری دارد.



شکل (۸): مقایسه F-measure روش ارایه شده با سایر روش‌ها در دیتاست PROMISE

برای دیتاست ناسا هم می‌توان دو پارامتر F-mesaure و AUC را محاسبه کرد. لازم به یادآوری است که پروژه JM1 بعنوان داده آموزشی بوده و مدل یادگیری بر اساس آن ساخته شده است. در جدول ۱۰ پارامتر F-measure و مقایسه آن برای چهار پروژه دیتاست NASA با روش‌های قبلی به دست آمده است.

جدول ۱۰ : مقایسه F-measure برای داده های NASA

نام پروژه	PC1	CM1	KC1	KC2
F-measure(%)				
RF	۶۸	۶۷	۶۲	۷۱
NB	۵۸	۵۸	۶۱	۵۵
DT	۵۷	۵۷	۶۰	۶۰
RT	۶۵	۶۴	۶۲	۶۲
MP	۶۴	۶۵	۶۶	۶۸
[۵]	۷۰	۷۲	۷۱	۷۱
[۳۳]	۶۸	۶۹	۶۳	۶۶
Proposed LR with all features	**۸۰**	۷۱	۶۵	۷۴
Proposed LR with five features	۷۵	**۷۷**	**۸۰**	**۷۶**

با توجه به جدول ۱۰ مشاهده می‌شود که F-measure ارایه شده برای چهار پروژه دیتاست NASA در روش ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است. بعنوان مثال در پروژه KC1 پارامتر F-measure بدست آمده برای روش ارایه شده در این مقاله برابر ۸۰٪ است که نسبت به روش جنگل تصادفی (۶۲٪)، درخت تصمیم (۶۰٪)، روش بیزین (۶۱٪)، درخت تصادفی (۶۲٪) ، پرسپترون چند لایه (۶۶٪) و روش ارایه شده در [۵] برابر ۷۱ ٪ و روش ارایه شده در [۳۳] برابر ۶۳ ٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر ۶۵ ٪ است. لازم به ذکر است که در حالت کلی و با در نظر گرفتن تمامی پروژه‌های این دیتاست روش LR مبتنی بر انتخاب پنج ویژگی مهم نتایج بهتری در مقایسه با سایر روش‌ها دارد. در جدول ۱۱ پارامتر AUC و مقایسه آن برای چهار پروژه دیتاست NASA با سایر روش‌ها به دست آمده است.

جدول ۱۱: مقایسه AUC برای داده های NASA

نام پروژه	PC1	CM1	KC1	KC2
AUC(%)				
RF	۶۸	۷۰	۶۰	۶۹
NB	۵۷	۵۸	۶۰	۵۶
DT	۵۶	۵۹	۶۳	۶۱
RT	۶۷	۶۶	۶۵	۶۵
MP	۶۲	۶۲	۶۸	۶۹
[۵]	۶۸	۷۱	۶۹	۶۸
[۳۳]	۷۰	۷۴	**۷۵**	۷۱
Proposed LR with all features	**۸۵**	۷۴	۷۲	۷۸
Proposed LR with five features	۷۶	**۸۰**	۷۴	**۸۳**

با توجه به جدول ۱۱ مشاهده می‌شود که AUC ارایه شده برای چهار پروژه دیتاست NASA در روش ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است. بعنوان مثال در پروژه PC1 پارامتر AUC بدست آمده برای روش ارایه شده در این مقاله برابر ۷۶٪ است که نسبت به روش جنگل تصادفی (۶۶٪)، درخت تصمیم (۵۶٪)، روش بیزین (۵۷٪)، درخت تصادفی (۶۷٪) ، پرسپترون چند لایه (۶۲٪) و روش ارایه شده در [۵] برابر ۶۸ ٪ و روش ارایه شده در [۳۳] برابر ۷۰ ٪ و با روش LR با در نظر گرفتن تمامی ویژگی‌ها برابر ۸۵ ٪ است لازم به ذکر است که در حالت کلی و با در نظر گرفتن تمامی پروژه‌های این دیتاست، روش LR مبتنی بر انتخاب پنج ویژگی مهم نتایج بهتری در مقایسه با سایر روش‌ها دارد.

علاوه بر انتخاب پروژه‌ها در هر یک از دیتاست‌ها، ما در این مقاله ارزیابی نوع دوم را بررسی می‌کنیم. در این حالت یکی از دیتاست‌ها را برای ساخت مدل یادگیری انتخاب کردیم و با دیتاست دیگر مدل را مورد ارزیابی قرار دادیم. این عمل را برای هر دو دیتاست تکرار کردیم. لازم به ذکر است که ویژگی‌های مشترک و نزدیک به هم بین دو دیتاست بعنوان ویژگی‌های مورد ارزیابی قرار گرفت. در جدول ۱۲ ویژگی‌های مشترک این دو دیتاست نشان داده شده است.

جدول ۱۲: ویژگی‌های مشترک دو دیتاست NASA و PROMISE

نام دیتاست	ویژگی‌های مشترک			
PROMISE	wmc	max-cc	loc	amc
NASA	D	V(G)	L.O.C	eV(G)

جدول ۱۲ چهار ویژگی مشترک و نزدیک به هم بین دو دیتاست را نشان می‌دهد. ابتدا دیتاست NASA به عنوان داده آموزشی در نظر گرفته شده است و دیتاست PROMISE به عنوان داده تست در نظر گرفته شده است. نتایج حاصل در جدول ۱۳ برای پارامترهای F-measure و AUC نشان داده شده است.

جدول۱۳ : F-measure و AUC - آموزش با مجموعه داده NASA، تست با مجموعه داده PROMISE

	F-measure	AUC
RF	۵۸	۶۰
NB	۵۲	۵۳
DT	۵۳	۵۲

RT	۵۵	۵۶
MP	۶۱	۵۸
[۵]	۶۲	۶۰
[۳۳]	۶۳	۶۷
Proposed LR with four features	۷۶	**۷۸**

با توجه به جدول ۱۳ مشاهده می‌شود که نتایج AUC و F-measure ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است. در این جدول دیتاست NASA به عنوان داده آموزشی در نظر گرفته شده است و دیتاست PROMISE به عنوان داده تست در نظر گرفته شده است. بعنوان مثال AUC بدست آمده در این مقاله برابر ۷۸ درصد است که نسبت به سایر روش‌ها نتایج بهتری دارد.

جدول۱۴ : مقایسه F-measure و AUC - آموزش با مجموعه داده PROMISE، تست با مجموعه دادهNASA

	F-measure	AUC
RF	۶۲	۶۶
NB	۵۸	۵۹
DT	۵۶	۵۵
RT	۶۰	۶۴
MP	۶۶	۶۹
[۵]	۷۱	۷۳
[۳۳]	۷۵	۷۴
Proposed LR with four features	۸۱	**۸۴**

نتایج حاصل از زمانیکه مدل با دیتاست PROMISE آموزش داده شد و با دیتاست NASA تست گردید در جدول ۱۴ نشان داده شده است. با توجه به جدول ۱۴ مشاهده می‌شود که AUC و F-measure ارایه شده در این مقاله نسبت به سایر روش‌ها بهتر است.. بعنوان مثال F-measure بدست آمده در این مقاله برابر ۸۱ درصد است که نسبت به سایر روش‌ها نتایج بهتری دارد.

۴-۵. بحث و بررسی نتایج ارزیابی‌ها

ما از دو دیتاست PROMISE و NASA برای ارزیابی روش ارایه شده در این مقاله استفاده کردیم. هر کدام از این دیتاست‌ها شامل پروژه‌های مختلفی هستند. که ما برای هر دیتاست، این پروژه‌ها را به دلیل میزان استفاده در سایر مقالات و داشتن نسبت عدم تعادل بین تعداد نمونه‌های خطا به نمونه‌های غیر خطا انتخاب کردیم. هر مجموعه داده حاوی چندین ویژگی است که آخرین ویژگی هر مجموعه داده خطا دار بودن یا عدم خطادار بودن را نشان می‌دهد. این دو دیتاست در سه حالت مختلف برای ارزیابی مورد استفاده قرار گرفتند. در حالت اول هر پروژه از هر دیتاست به سه بخش داده آموزشی، اعتبارسنجی و تست تقسیم شدند که با داده‌های آموزشی و اعتبارسنجی مدل یادگیری ساخته شد و با داده های تست عملکرد روش ارایه شده مورد ارزیابی قرار گرفت. این روش در بیشتر روش‌های ارایه شده [۵,۳۳] استفاده شدند. در حالت دوم یکی از پروژه‌های هر دیتاست که تعداد نمونه‌های آن بیشتر باشد بعنوان داده آموزشی و اعتبارسنجی در نظر گرفته می‌شود و داده‌های پروژه های دیگر همان دیتاست بعنوان داده تست مورد ارزیابی قرار گرفته و عملکرد سیستم سنجیده می‌شود. برای مجموعه داده PROMISE ما پروژه PROP را بعنوان داده آموزشی در نظر گرفتیم همچنین از دیتاست NASA پروژه JM1 را بعنوان داده آموزشی و اعتبارسنجی در نظر می‌گیریم. دلیل انتخاب این دو پروژه از دو دیتاست حجم داده زیاد آنها می‌باشد تا مدل یادگیری بتواند مدل کامل و جامعی ایجاد کند. این روش در [۵] استفاده شده است و در [۳۳] استفاده نشده است. در حالت سوم یک دیتاست بعنوان داده آموزشی و اعتبارسنجی در نظر گرفته می‌شود و مدل یادگیری بر اساس آن دیتاست ایجاد می‌شود و داده-های دیتاست دیگر بعنوان داده تست عملکرد سیستم را مورد ارزیابی قرار می-دهد. این روش در این مقاله برای نشان دادن تعمیم پذیری روش ارایه شده بکار گرفته شده است و در در [5,33] استفاده نشده است. دلیل استفاده کمتراین روش این است که در این حالت باید ابتدا ویژگی‌های مشترک دو دیتاست محاسبه شده و سپس ارزیابی‌ها انجام گردد.

از طرف دیگر مدل‌های استاندارد پیش‌بینی خطا دارای محدودیت‌هایی مانند دقت پایین هستند [۵]. تنوع زیاد مقادیر ویژگی‌ها، نامتعادل بودن تعداد داده-های بدون خطا و خطا در دیتاست‌ها و یکسان در نظر گرفتن وزن هر ویژگی باعث کاهش نتایج دقت، F-measure و AUC می‌شود. الگوریتم‌های طبقه‌بندی زمانی که تنوع مقادیر ویژگی کمتر باشد، نتایج بهتری تولید می‌کنند. ما در این مقاله با الگوریتم دانه‌بندی تنوع مقادیر داده‌ها را به سه کلمه تقلیل دادیم. هم چنین با در نظر گرفتن اهمیت ویژگی‌ها، عملکرد پارامترهای ارزیابی را بهبود دادیم. در نهایت با متعادل سازی نسبت داده‌ای خطا به غیر خطا، توزیع داده‌های خطا و بدون خطا اصلاح شد. نتایج این مقاله در سه حالت ارزیابی بر روی هر دو مجموعه داده نشان می‌دهد که روش ارایه شده در این مقاله عملکرد بهتری دارد.

۵- مقایسه و نتیجه‌گیری

در این مقاله برای پیش بینی خطاهای نرم افزار از یک روش مبتنی بر دانه بندی داده‌ها استفاده شده است در این مقاله ابتدا بر اساس الگوریتم ارایه شده در این مقاله مقادیر دانه بندی شدند. سپس با استفاده از مقادیر حاصل اهمیت ویژگی‌ها محاسبه شدند. سپس بر اساس الگوریتم oversampling مساله عدم تعادل برای متناسب نگه داشتن نسبت داده‌های خطا و بدون خطا بر۳ای داده‌های آموزشی انجام شد. در ادامه پارامترهای دقت و F-measure و AUC برای دو دیتاست PROMISE و NASA محاسبه شد. همچنین نتایج این مقاله با شش روش یادگیری ماشین و روش‌های قبلی مقایسه شده است و نشان داده شد که روش ارائه شده بهتر از سایر روش‌ها می‌باشد. در این مقاله سه نوع ارزیابی صورت پذیرفت. در حالت اول برای هر پروژه از هر دیتاست که داده آموزشی و اعتبارسنجی و تست از همان پروژه انتخاب شده بود، انجام شد. در حالت دوم یکی از پروژه های هر دیتاست بعنوان داده آموزشی و اعتبارسنجی در نظر گرفته شد و پروژه های دیگر همان دیتاست بعنوان داده تست در نظر گرفته شد. در مرحله سوم داده های یک دیتاست بعنوان داده آموزشی و اعتبارسنجی در نظر گرفته شد و داده های دیتاست دیگر بعنوان تست در نظر گرفته شد. نتایج این مقاله برای شرکت‌های تازه تاسیس و نوپا که می‌خواهند از تجربیات و دیتاست‌های شرکت‌های بزرگ برای پیش‌بنی خطاهای محصول-شان استفاده کنند، بسیار سودمند می‌باشد. از جمله کارهایی که می‌توان در آینده انجام داد این است که ابتدا می‌توان الگوریتم دانه‌بندی را بهبود داد تا بتواند بهترین بازه هر ویژگی را پیدا کند تا بتوان دقت روش ارایه شده را بالاتر برد.

مراجع

1- Krasner, H. (2020). The cost of poor quality software in the us: A 2018 report, 2018.

2- Rathore, S. S., & Kumar, S. (2017). Towards an ensemble based system for predicting the number of software faults. *Expert Systems with Applications*, *82*, 357-382. https://doi.org/10.1016/j.eswa.2017.04.014.

3- Tufano, M., Watson, C., Bavota, G., Di Penta, M., White, M., & Poshyvanyk, D. (2018, September). An empirical investigation into learning bug-fixing patches in the wild via neural machine translation. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (pp. 832-837). https://doi.org/10.1145/3238147.3240732.

4- Turabieh, H., Mafarja, M., & Li, X. (2019). Iterated feature selection algorithms with layered recurrent neural network for software fault prediction. *Expert systems with applications*, *122*, 27-42. https://doi.org/10.1016/j.eswa.2018.12.033.

5- Juneja, K. (2019). A fuzzy-filtered neuro-fuzzy framework for software fault prediction for inter-version and inter-project evaluation. *Applied Soft Computing*, *77*, 696-713. https://doi.org/10.1016/j.asoc.2019.02.008.

6- Bénard, C., Biau, G., Da Veiga, S., & Scornet, E. (2021). Sirus: Stable and interpretable rule set for classification. *Electronic Journal of Statistics*, *15*(1), 427-505. https://doi.org/10.1214/20-EJS1792.

7- Bénard, C., Biau, G., Da Veiga, S., & Scornet, E. (2021, March). Interpretable random forests via rule extraction. In *International Conference on Artificial Intelligence and Statistics* (pp. 937-945). PMLR. http://dx.doi.org/10.48550/arXiv.2004.14841.

8- Ma, B., Dejaeger, K., Vanthienen, J., & Baesens, B. (2011). Software defect prediction based on association rule classification. Available at SSRN 1785381. http://dx.doi.org/10.2139/ssrn.1785381.

9- Elish, K. O., & Elish, M. O. (2008). Predicting defect-prone software modules using support vector machines. Journal of Systems and Software, 81(5), 649-660. https://doi.org/10.1016/j.jss.2007.07.040.

10- Thwin, M. M. T., & Quah, T. S. (2005). Application of neural networks for software quality prediction using object oriented metrics. Journal of systems and software, 76(2), 147-156. https://doi.org/10.1016/j.jss.2004.05.001.

11- Turhan, B., & Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study.Data & Knowledge Engineering, 68(2), 278-290. https://doi.org/10.1016/j.datak.2008.10.005.

12- Caglayan, B., Tosun Misirli, A., Bener, A. B., & Miranskyy, A. (2015). Predicting defective modules in different test phases. Software Quality Journal, 23(2), 205-227. https://doi.org/10.1007/s11219-014-9230-x.

13- Mohanta, S., Vinod, G., Ghosh, A. K., & Mall, R. (2010). An approach for early prediction of software reliability. ACM SIGSOFT Software Engineering Notes, 35(6), 1-9. https://doi.org/10.1145/1874391.1874403.

14- Mohanta, S., Vinod, G., & Mall, R. (2011). A technique for early prediction of software reliability based on design metrics. International Journal of System Assurance Engineering and Management, 2(4), 261-281. https://doi.org/10.1007/s13198-011-0078-1.

15- Okutan, A., & Yıldız, O. T. (2014). Software defect prediction using Bayesian networks. Empirical Software Engineering, 19(1), 154-181. https://doi.org/10.1007/s10664-012-9218-8.

16- Dejaeger, K., Verbraken, T., & Baesens, B. (2012). Toward comprehensible software fault prediction models using bayesian network classifiers. IEEE Transactions on Software Engineering, 39(2), 237-257. https://doi.org/10.1109/TSE.2012.20.

17- Catal, C., & Diri, B. (2009). Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Information Sciences, 179(8),1040-1058. https://doi.org/10.1016/j.ins.2008.12.001.

18- Rathore, S. S., & Kumar, S. (2017). Towards an ensemble based system for predicting the number of software faults. Expert Systems with Applications, 82, 357-382. https://doi.org/10.1016/j.eswa.2017.04.014.

19- Jin, C., Jin, S. W., & Ye, J. M. (2012). Artificial neural network-based metric selection for software fault-prone prediction model. I.E.T. software, 6(6), 479-487. https://doi.org/10.1049/iet-sen.2011.0138.

20- Venugopal, K., Madhusudan, P., & Amrutha, A. (2017, July). Artificial neural network based fault prediction framework for transformers in power systems. In 2017 International Conference on Computing Methodologies and Communication (ICCMC) (pp. 520-523). IEEE. https://doi.org/10.1109/ICCMC.2017.8282519.

21- Kamei, Y., & Shihab, E. (2016, March). Defect prediction: Accomplishments and future challenges. In 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER) (Vol. 5, pp. 33-45). IEEE. https://doi.org/10.1109/SANER.2016.56.

22- Fenton, N., Neil, M., Marsh, W., Hearty, P., Radliński, Ł., & Krause, P. (2008). On the effectiveness of early life cycle defect prediction with Bayesian Nets. Empirical Software Engineering, 13(5), 499-537. https://doi.org/10.1007/s10664-008-9072-x.

23- Pandey, A. K., & Goyal, N. K. (2013). Multistage model for residual fault prediction. In Early software reliability prediction (pp. 59-80). Springer, India. https://doi.org/10.1007/978-81-322-1176-1_4.

24- Ying, M., Zhu, S., & Ke, Q. (2014). Combining the requirement information for software defect estimation in design time [J]. Information Processing Letters, 114(9), 469-474. https://doi.org/10.1016/j.ipl.2014.03.012.

25- Yadav, D. K., Chaturvedi, S. K., & Misra, R. B. (2012). Xde . International Journal of Performability Engineering, 8(4). https://doi.org/10.23940/ijpe.12.4.p399.mag.

26- Yadav, H. B., & Yadav, D. K. (2015). A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Information and Software Technology, 63, 44-57. https://doi.org/10.1016/j.infsof.2015.03.001.

27- Ebru, A., & Parvinder, S. S. (2010). A soft computing approach for modeling of severity of faults in software systems. International Journal of Physical Sciences, 5(2), 74-85. https://doi.org/10.1007/978-81-322-1176-1_4.

28- Chatterjee, S., & Roy, A. (2015). Novel algorithms for web software fault prediction. Quality and Reliability Engineering International, 31(8), 1517-1535. https://doi.org/10.1002/qre.1687.

29- Chatterjee, S., Nigam, S., & Roy, A. (2017). Software fault prediction using neuro-fuzzy network and evolutionary learning approach. Neural Computing and Applications, 28(1), 1221-1231.https://doi.org/10.1007/s00521-016-2437-y.

30- Sharma, D., & Chandra, P. (2019). A comparative analysis of soft computing techniques in software fault prediction model development. International Journal of Information Technology, 11(1), 37-46. https://doi.org/10.1007/s41870-018-0211-3.

31- Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2021). Machine learning based methods for software fault prediction: A survey. Expert Systems with Applications, 172, 114595. https://doi.org/10.1016/j.eswa.2021.114595.

32- Yang, X., Yu, H., Fan, Guisheng, Y. K. (2020). A differential evolution-based approach for effort-aware just-in-time software defect prediction. 13-16. https://doi.org/10.1145/3416506.3423577.

33- Jin, C. (2021). Software defect prediction model based on distance metric learning. Soft Computing, 25(1), 447-461. https://doi.org/10.1007/s00500-020-05159-1.

34- Shirabad, J. S., & Menzies, T. J. (2005). The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada. URL http://promise. site. uottawa. ca/SERepository.

35- NASAIV&Vfacility,Metric data program, available from http://MDP.ivv.nasa.gov/.

36- Kaburlasos VG, Tsoukalas V, Moussiades L (2014) Fcknn: a granular knn classifier based on formal concepts. In: 2014 IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 61–68. https://doi.org/10.1109/FUZZ-IEEE.2014.6891726.

37- Pedrycz, Witold. (2013). Granular Computing: Analysis and Design of Intelligent Systems (1st E.). Taylor & Francis, Year. https://doi.org/10.1201/9781315216737.

38- Kamalov, F., Thabtah, F. (2017). A Feature Selection Method Based on Ranked Vector Scores of Features for Classification. *Ann. Data. Sci.* **4**, 483–502 https://doi.org/10.1007/s40745-017-0116-1.

Behrooz shahi is a research assistant and pursuing a PhD degree in the department of Computer Engineering at Shiraz University. His research interests include all aspects of software quality and are focused on software fault prediction models.

Hooman Tahayori is an Associate professor and a full time faculty member in the Department of Computer Engineering at Shiraz University. Her research interests are Perceptual computing, Fuzzy Logic, and software defect prediction.

Software Fault Prediction through Information Granulation

Behrooz Shahi, Hooman Tahayori

C.S.E. and I.T. Department, Shiraz University, Shiraz 71847-51541, Iran

Email: b.shahi@shirazu.ac.ir, tahayori@shirazu.ac.ir

Abstract:

Software systems are complex. Time and cost limits make it hard to find all faults in them during development. Therefore, it is necessary to predict faults using previous versions of projects or other projects in order to have software with high reliability. Software fault prediction algorithms lack generalizability. They also suffer from an imbalance between faulty and clean data. And, the values of features vary between projects. These are the basic challenges in these systems. This paper presents a four-level architecture for predicting software errors. First, an algorithm for granularity of numerical values is introduced. Then the importance of the features is calculated based on the granular values. Then, the data is divided into three parts: training, validation, and testing. The oversampling algorithm is used to balance the ratio of faulty data to clean data on the training data. At the end, evaluation and comparison with other methods is done. PROMISE and NASA datasets have been used to evaluate the proposed method. Finally, three types of evaluation and comparison with other methods are done using PROMISE and NASA datasets. The results of this article show that fault prediction in all three types of evaluation in most projects of each dataset has a better performance than other methods. It is also shown that the generalizability presented in this article can be done on any dataset.

Keywords: Software reliability, Software fault prediction, Information granularity. Generalizability, machine learning