تأثیر ترکیب روش‌های انتخاب ویژگی فیلتر و بسته‌بندی در بهبود پیش‌بینی اشکال نرم‌افزار

فاطمه علیقارداشی[۱]، دانشجوی کارشناسی ارشد؛ محمدعلی زارع چاهوکی[۲]، استادیار

۱- دانشکده مهندسی برق و کامپیوتر - دانشگاه یزد - یزد - ایران - f.alighardashi@stu.yazd.ac.ir
۲- دانشکده مهندسی برق و کامپیوتر - دانشگاه یزد - یزد - ایران - chahooki@yazd.ac.ir

چکیده: حفظ کیفیت محصول نرم‌افزاری با آزمون‌های دوره‌ای قبل از نصب، یکی از پرهزینه‌ترین فعالیت‌ها در پروژه‌های فناوری اطلاعات است. با توجه به منابع محدود برای آزمون ماژول‌ها در پروژه‌های نرم‌افزاری، بهتر است ابتدا ماژول‌های مستعد اشکال شناسایی شوند و منابع آزمون در جهت شناسایی اشکال در این ماژول‌ها متمرکز گردند. پیش‌بینی‌کننده‌های اشکال مبتنی بر الگوریتم‌های یادگیری ماشین، ابزارهای مقرون‌به‌صرفه‌ای برای شناسایی ماژول‌های مستعد اشکال هستند. پژوهش‌های گسترده‌ای در این حوزه برای یافتن ارتباط بین ویژگی‌های ماژول‌های نرم‌افزاری و مستعد اشکال بودن آن‌ها صورت پذیرفته است. برخی از این ویژگی‌ها در الگوریتم‌های پیش‌بینی‌کننده به‌گونه‌ای هستند که نه‌تنها سبب بهبود دقت در فرآیند یادگیری نمی‌شوند بلکه کاهش دقت را نیز در پی خواهند داشت. در این پژوهش با توجه به عملکرد خوب روش انتخاب ویژگی روبه‌جلو در انتخاب ویژگی‌های مؤثر، زیرمجموعه اولیه در این روش با استفاده از تلفیق ویژگی‌های با رتبه بالا در روش‌های مختلف فیلتر انتخاب می‌شود. روش پیشنهادی علاوه‌بر بهبود دقت سبب افزایش سرعت همگرایی در انتخاب ویژگی می‌شود. نتایج حاصل از پیاده‌سازی و ارزیابی نتایج تجربی به‌دست‌آمده در دادگان ناسا با معیار AUC، بیانگر مؤثر بودن روش پیشنهادی در بهبود دقت و سرعت پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکال است.

واژه‌های کلیدی: پیش‌بینی اشکال نرم‌افزار، یادگیری ماشین، انتخاب ویژگی، فیلتر، بسته‌بندی.

The Effectiveness of the Combination of Filter and Wrapper Feature Selection Methods to Improve Software Fault Prediction

F. Alighardashi[1], MSc Student; M. A. Zare Chahooki[2], Assistant Professor

1- Faculty of Electrical and Computer Engineering, University of Yazd, Yazd, Iran, Email: f.alighardashi@stu.yazd.ac.ir
2- Faculty of Electrical and Computer Engineering, University of Yazd, Yazd, Iran, Email: chahooki@yazd.ac.ir

Abstract: Improving the software product quality before releasing by periodic tests is one of the most expensive activities in software projects. Due to limited resources to test modules in software projects, it is important to identify fault-prone modules and use the test sources for fault prediction in these modules. Software fault predictors based on machine learning algorithms, are effective tools for identifying fault-prone modules. Extensive studies are being done in this field to find the connection between features of software modules, and fault-prone have been done. Some of these features in predictive algorithms are like that not only did not improve the accuracy of the learning process, but also will be reduced the accuracy. In this study, due to the excellent performance of Forward Feature Selection (FS) method for effective selection of features, the initial subset of this method has been selected by using of combination of high ranking features in different Filter methods. The proposed method causes increment the speed of the convergence of feature selection as well as the accuracy improvement. The obtained results on NASA dataset with AUC criteria, indicates the effectiveness of this method in the improvement of the accuracy and the speed of software fault prediction.

Keywords: Software fault prediction, machine learning, feature selection, filter, wrapper.

تاریخ ارسال مقاله: ۱۳۹۴/۱۱/۱۸
تاریخ اصلاح مقاله: ۱۳۹۵/۰۱/۲۹ و ۱۳۹۵/۰۲/۲۲
تاریخ پذیرش مقاله: ۱۳۹۵/۰۳/۱۱
نام نویسنده مسئول: محمدعلی زارع چاهوکی
نشانی نویسنده مسئول: ایران - یزد - بلوار دانشگاه - دانشگاه یزد - دانشکده مهندسی برق و کامپیوتر.

۱- مقدمه

در طول سال‌های اخیر درخواست برای کیفیت نرم‌افزار رشد بالایی داشته است. همچنین با توجه به پیچیده‌تر شدن سیستم‌های نرم‌افزاری، پیچیدگی مسائل مرتبط به آزمون نرم‌افزار نیز به‌طور فزاینده‌ای افزایش یافته است. قابلیت اندازه‌گیری مستعد اشکال بودن ماژول‌های نرم‌افزاری می‌تواند به‌شدت روی کاهش هزینه و بهبود فرآیند آزمون نرم‌افزار تأثیرگذار باشد. اکثر خطاهای سیستم نرم‌افزاری از تعداد اندکی از اجزای سیستمی ناشی می‌شوند. قانون ۸۰:۲۰ در این زمینه بیانگر آن است که ۲۰ درصد از ماژول‌های نرم‌افزاری سبب ایجاد ۸۰ درصد خطا، هزینه و کار مجدد می‌شود. بنابراین با تلاش برای رفع خطا و بهبود کیفیتِ بخش کوچکی از نرم‌افزار، اتلاف زمان کمتری در کل پروسه تولید خواهیم داشت [۱].

با توجه به منابع محدود برای آزمون ماژول‌ها در پروژه‌های نرم‌افزاری، بهتر است ابتدا ماژول‌های مستعد اشکال (fp[۱]) شناسایی شده و از ماژول‌های غیرمستعد اشکال (nfp[۲]) تفکیک شوند و منابع آزمون در جهت شناسایی اشکال در ماژول‌های مستعد اشکال متمرکز شود. این مسئله موجب مطرح شدن مبحث پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکال شد که به‌اصطلاح «پیش‌بینی اشکال نرم‌افزار» نیز گفته می‌شود [۱].

پیش‌بینی اشکال نرم‌افزار، مدیران پروژه‌های نرم‌افزاری را قادر می‌سازد که تلاش برای بهبود کیفیت نرم‌افزار را در قسمت‌های موردنیاز متمرکز کنند. به‌عنوان‌مثال، قبل از آزمون سیستم، شناسایی اجزایی که به احتمال زیاد در حین عملیات باعث بروز خطا می‌گردند، می‌تواند اثربخشی تلاش برای آزمون نرم‌افزار را بهبود بخشد. به همین دلیل روش‌های مختلف مدل‌سازی پیش‌بینی اشکال نرم‌افزار توسعه داده شده‌اند [۲].

پیش‌بینی‌کننده‌های اشکال مبتنی بر الگوریتم‌های یادگیری ماشین، ابزارهای مقرون‌به‌صرفه‌ای برای شناسایی ماژول‌های نرم‌افزاری مستعد اشکال هستند. پیش‌بینی اشکال نرم‌افزار مبتنی بر یادگیری ماشین رویکردی است که از ویژگی‌های نرم‌افزاری پروژه‌های پیشین استفاده کرده سپس مدلی ایجاد می‌کند که ماژول‌های مستعد اشکال نسخه بعدی نرم‌افزار را پیش‌بینی می‌کند [۳].

مدل مستعد اشکال بر اساس اطلاعات کدهای نرم‌افزاری و وجود یا عدم وجود اشکال در آن‌ها ساخته شده است. اطلاعات اشکال به‌عنوان متغیر وابسته در مدل‌های پیش‌بینی اشکال نرم‌افزار در نظر گرفته می‌شود. اگر خطایی در حین انجام آزمایش سیستم نرم‌افزاری گزارش شود، اطلاعات اشکال ماژول با مقدار غیرصفر مقداردهی می‌شود و در غیراین‌صورت مقدار صفر یعنی بدون اشکال به آن نسبت داده می‌شود. برای مدل‌سازی پیش‌بینی، ویژگی‌های نرم‌افزار به‌عنوان متغیر مستقل در مدل پیش‌بینی استفاده شده و اطلاعات اشکال به‌عنوان متغیر وابسته یا هدف در مدل پیش‌بینی استفاده می‌شود [۴].

مدل‌های پیش‌بینی اشکال نرم‌افزار با شروع از سال ۱۹۹۰ تا به امروز توسط پژوهشگران بسیاری موردمطالعه قرار گرفته‌اند که نتیجه آن شناسایی ماژول‌های نرم‌افزاری مستعد اشکال قبل از آزمون سیستم با استفاده از این مدل‌ها بوده است. در پژوهش [۵]، منزیس و همکارانش میزان تشخیص اشکال در ماژول‌های نرم‌افزاری توسط مدل‌های پیش‌بینی را حدود ۷۱ درصد اعلام کردند که بالاتر از میزان تشخیص اشکال توسط منابع انسانی (۶۰ درصد) بوده است. ایشان در جلساتی که کنفرانس متریک IEEE در سال ۲۰۰۲ برگزار کرد، ادعای فاگان مبنی بر این‌که بررسی کد نرم‌افزار قبل از فرآیند آزمون، توسط منابع انسانی می‌تواند ۹۵ درصد از نقایص نرم‌افزار را برطرف کند را رد کرد. چراکه بر اساس پژوهش مذکور ثابت شد که نسبت تشخیص اشکال‌های کد نرم‌افزاری توسط منابع انسانی حدود ۶۰ درصد می‌شود. منزیس در این راستا بیان کرد که هر یک از اعضای تیم بررسی کد نرم‌افزار می‌توانند حدود ۸ تا ۲۰ خط کد را در هر دقیقه بررسی کنند و آن‌ها فرآیند بررسی را به‌وسیله تمام اعضای تیم متشکل از ۴ تا ۶ نفر انجام داده‌اند. اما بررسی کد بسیار زمان‌بر بوده و تنها حدود ۶۰ درصد از اشکال‌های موجود در ماژول‌های نرم‌افزاری تشخیص داده شده‌اند. بدین سبب، ثابت شد که استفاده از مدل‌های پیش‌بینی اشکال نرم‌افزار در مقایسه با بررسی نرم‌افزار توسط تیم نرم‌افزاری، بسیار مقرون‌به‌صرفه‌تر است [۴].

انتخاب ویژگی، یکی از رویکردهای بهبود دقت و سرعت در روش‌های یادگیری ماشین است [۶]. در سال‌های اخیر پژوهش‌های متعددی در حوزه انتخاب ویژگی در پیش‌بینی اشکال نرم‌افزار صورت پذیرفته است [۷]. پژوهش‌های مذکور نشان‌دهنده تعداد بالای ویژگی‌های استخراج‌شده از ماژول‌های نرم‌افزاری هستند. نتایج پژوهش‌ها در حوزه کاهش ویژگی‌ها نشان داده است که انتخاب مجموعه‌ای کمتر از ویژگی‌های اولیه می‌تواند سبب افزایش دقت در الگوریتم‌های یادگیری ماشین شود. پیش‌بینی اشکال نرم‌افزار با استفاده از لیست کاملی از ویژگی‌های در دسترس، اغلب سبب کاهش دقت در پیش‌بینی می‌شود. چراکه در برخی موارد ممکن است که برخی از این ویژگی‌ها نامربوط یا افزونه باشند و سبب ایجاد سردرگمی مدل یادگیری و نیز پیچیدگی‌های اضافی در فرآیند ساخت مدل پیش‌بینی توسط روش یادگیری ماشین شوند. برای مقابله با این مسئله، از انتخاب ویژگی استفاده می‌شود. انتخاب ویژگی فرایند یافتن زیرمجموعه‌ای از ویژگی‌هاست که با آن‌ها یک مدل، دقتی مشابه یا بهتر نسبت به استفاده از تمام ویژگی‌ها دارد [۷].

یکی از مشکلات دادگان‌های ارزیابی این است که در بیش‌تر مواقع تمام ویژگی‌های دادگان برای یافتن دانشی که در آن‌ها نهفته است مهم و حیاتی نیستند، این مسئله در پژوهش‌های بسیاری [۱۰-۸] به‌عنوان فلسفه اصلی نیاز به کاهش ویژگی‌های دادگان برای ایجاد مدل یادگیری بهتر و بهبود دقت مدل بر دادگان بیان شده است. چراکه روش‌های یادگیری ماشین، با استفاده از ارتباط‌های بین ویژگی‌های

تشریح‌کننده نمونه‌های دادگان آموزشی، الگویی را استخراج می‌کنند و با استفاده از این الگو و ویژگی‌های تشریح‌کننده نمونه جدید، پیش‌بینی اشکال نرم‌افزار را برای این نمونه انجام می‌دهند [۷]. بنابراین، بی‌شک ویژگی‌های تشریح‌گر نمونه‌ها تأثیر فوق‌العاده‌ای روی دقت روش‌های یادگیری ماشین دارند و دقت و درستی مدل مورداستفاده، به ویژگی‌های مجموعه دادگان بستگی دارد. بنابراین نیاز است که بهترین ویژگی‌ها برای ایجاد مدل بهتر توسط روش‌های انتخاب ویژگی انتخاب شوند.

الگوریتم انتخاب ویژگی می‌تواند تلفیقی از تکنیک‌های جستجو برای ارائه زیرمجموعه‌ای جدید از ویژگی‌ها، همراه با معیاری برای ارزیابی که زیرمجموعه‌های متمایز را امتیازدهی می‌کند، باشد [۱۱]. رویکردی که اخیراً در حوزه انتخاب ویژگی مطرح شده است، ترکیب روش‌های انتخاب ویژگی است. ایده ترکیب روش‌های انتخاب ویژگی برای کاربردهای دیگر [۱۴-۱۲] نیز مطرح شده است. نتایج پژوهش‌های اخیر بیانگر تأثیر مثبت روش‌های ترکیبی انتخاب ویژگی در کاربردهای حوزه مهندسی نرم‌افزار ازجمله پیش‌بینی اشکال نرم‌افزار نیز بوده است [۱۵، ۱۶]. بنابراین فرضیه‌ای که در این پژوهش مطرح شده است، آن است که استفاده از این روش‌ها در تشخیص ماژول‌های نرم‌افزاری مستعد اشکال می‌تواند دقت پیش‌بینی و سرعت فرآیند یادگیری روش‌های یادگیری ماشین را بهبود بخشد. در این راستا، روش انتخاب ویژگی جدیدی برای بهبود فرآیند پیش‌بینی اشکال نرم‌افزار در این پژوهش ارائه شده است.

در بین روش‌های بسته‌بندی، روش انتخاب روبه‌جلو (FS^3) [۶] عملکرد بسیار خوبی در انتخاب ویژگی‌ها دارد، اما این روش ازنظر سرعت بسیار کند است. سرعت پایین این روش به‌دلیل شروع به کار آن با یک مجموعه خالی از ویژگی‌ها و در ادامه اضافه کردن هر یک از ویژگی‌های مجموعه ویژگی‌ها به‌طور جداگانه به مجموعه اولیه است. ایده پیشنهادی ما، اصلاح مجموعه اولیه ویژگی‌های روش FS با استفاده از تلفیق ویژگی‌های با رتبه بالا در روش‌های فیلتر مختلف است. با توجه به سرعت خوب روش‌های فیلتر، مجموعه پیشنهادی ویژگی‌های اولیه برای شروع در اختیار روش FS قرار می‌گیرد. همچنین مجموعه ویژگی‌های اضافه‌شونده نیز از مابقی ویژگی‌ها تشکیل می‌شود. بنابراین در این پژوهش برای افزایش سرعت و دقت همگرایی روش‌های پایه انتخاب ویژگی، روشی دومرحله‌ای از تلفیق روش‌های فیلتر و بسته‌بندی ارائه شده است در اغلب پژوهش‌های صورت پذیرفته در حوزه پیش‌بینی اشکال نرم‌افزار، تنها روش‌های فیلتر مختلف بر دادگان اعمال و نتایج آن‌ها اعلام شده است. در این پژوهش روشی تلفیقی از چند روش فیلتر ارائه شده است که تاکنون در دیگر پژوهش‌ها چنین ترکیبی در تلفیق روش‌های فیلتر ارائه نشده است.

بدون ترکیب روش‌های فیلتر، نتایج پژوهش‌ها نشان‌دهنده آن است که روش‌های فیلتر مختلف بر دادگان مختلف متفاوت عمل کرده‌اند و هر روش فیلتر در برخی دادگان دقت پیش‌بینی بسیار بالا و در برخی دیگر دقت بسیار پایینی داشته‌اند. بنابراین نمی‌توان تعیین کرد که کدام روش فیلتر برای پیش‌بینی اشکال پروژه نرم‌افزاری جدید بهتر است. همان‌گونه که در نتایج آزمایش‌های تجربی نشان داده شده است، روش تلفیقی ارائه‌شده در این پژوهش در تمامی دادگان‌های آزمون‌شده مؤثر است. به‌عبارت‌دیگر نتایج به‌دست‌آمده از پیاده‌سازی روش پیشنهادی نشان‌دهنده بهبود دقت و سرعت نسبت به استفاده از تمامی ویژگی‌ها است.

همچنین هرچند تاکنون از SVM [۲] در پژوهش‌های مختلفی در حوزه پیش‌بینی اشکال نرم‌افزار استفاده شده است ولی از برخی از توسعه‌های جدیدی که در این فرضیه یادگیری داده شده است، در حوزه پیش‌بینی اشکال نرم‌افزار استفاده نشده است. نوآوری جنبی ارائه‌شده در این پژوهش، افزایش دقت دسته‌بندی با ماشین بردار پشتیبان دوقلو هموارشده با حاشیه پارامتریک (STPMSVM) [۱۷] است.

در ادامه این پژوهش در بخش ۲، روش‌های انتخاب ویژگی معرفی شده‌اند. در بخش ۳، پژوهش‌های مبتنی بر انتخاب ویژگی در پیش‌بینی اشکال نرم‌افزار مرور شده‌اند. در بخش ۴، روش پیشنهادی این پژوهش بیان شده است. در بخش ۵ و ۶ به‌ترتیب دادگان و معیارهای ارزیابی کارایی در حوزه پیش‌بینی اشکال مبتنی بر یادگیری ماشین آورده شده است. در بخش ۷ نیز نتایج تجربی آزمایش‌های صورت‌پذیرفته جهت بررسی تأثیر روش پیشنهادی بر پیش‌بینی ماژول‌های مستعد اشکال، آورده شده است و این نتایج با نتایج دیگر پژوهش‌های انجام‌شده در زمینه پیش‌بینی اشکال نرم‌افزار مقایسه و تحلیل شده است. درنهایت در بخش ۸ نیز نتیجه‌گیری و پژوهش‌های پیش رو آورده شده است.

۲- انتخاب ویژگی

کاهش ویژگی، یکی از پرکاربردترین روش‌هایی است که در مبحث یادگیری ماشین مطرح شده است. هدف کاهش ویژگی، حذف ویژگی‌های غیرضروری و انتخاب مناسب‌ترین ویژگی‌ها از بین مجموعه ویژگی‌های اولیه، برای افزایش عملکرد الگوریتم‌های یادگیری است [۱۸]. معمولاً جستجوی جامع برای پیدا کردن مناسب‌ترین ویژگی‌ها به‌لحاظ هزینه محاسباتی غیرممکن است. بنابراین کاهش ویژگی تبدیل به یک چالش عمده در شناسایی الگو و یادگیری ماشین شده است. این روش در بسیاری از کاربردها ازجمله طبقه‌بندی و رگرسیون اهمیت بسیاری دارد. چراکه در این کاربردها معمولاً تعداد زیادی ویژگی وجود دارد، که بسیاری از آن‌ها یا اثربخش نیستند و یا سبب کاهش دقت یادگیری می‌شوند. حذف کردن این ویژگی‌ها علاوه‌بر افزایش دقت، سبب کاهش پیچیدگی محاسباتی می‌شود [۱۹].

روش‌های انتخاب ویژگی، روش‌هایی مبتنی بر کاهش ویژگی هستند. این روش‌ها سعی می‌کنند با انتخاب زیرمجموعه‌ای از ویژگی‌های اولیه، ابعاد داده‌ها را کاهش دهند. در این روش‌ها به‌دنبال

پیدا کردن زیرمجموعه‌ای با حداقل اندازه ممکن از ویژگی‌ها هستیم که برای کاربرد موردنظر مناسب باشد. در اغلب موارد تحلیل‌های داده‌ای نظیر طبقه‌بندی بر روی فضای کاسته‌شده نسبت به فضای اصلی بهتر عمل می‌کند [۱۹].

روش‌های انتخاب ویژگی، تلاش می‌کنند تا از میان مجموعه‌ای با N ویژگی و 2^N زیرمجموعه کاندید، بهترین زیرمجموعه را پیدا کنند. در تمام این روش‌ها بر اساس کاربرد و نوع تعریف، زیرمجموعه‌ای به‌عنوان جواب انتخاب می‌شود، که بتواند مقدار یک تابع ارزیابی را بهینه کند. با وجود این‌که هر روش سعی می‌کند تا بهترین ویژگی‌ها را انتخاب کند، اما با توجه به وسعت جواب‌های ممکن، پیدا کردن جواب بهینه مشکل و در Nهای متوسط و بزرگ بسیار پرهزینه است [۱۹]. دو دسته‌بندی اصلی از روش‌های انتخاب ویژگی شامل روش‌های فیلتر و بسته‌بندی هستند. در ادامه این بخش این دو روش شرح داده می‌شوند.

۲-۱- روش فیلتر

روش انتخاب ویژگی فیلتر از معیاری کاندید برای امتیازدهی به زیرمجموعه ویژگی‌ها استفاده می‌کند. در این روش، از هیچ روش یادگیری ماشینی استفاده نمی‌شود. به‌عبارتی‌دیگر، در روش‌های فیلتر، مستقل از الگوریتم یادگیری ماشین، زیرمجموعه‌های ویژگی به‌وسیله مفاهیمی ازجمله اطلاعات ویژگی، فاصله، سازگاری، شباهت و دیگر معیارهای آماری انتخاب شده و شایستگی آن‌ها مورد ارزیابی قرار می‌گیرد. در این رویکرد با توجه به معیار کاندید، به هر یک از ویژگی‌ها امتیاز داده می‌شود و بر اساس آن ویژگی‌ها مرتب می‌شوند. این روش‌ها همچنین به‌عنوان پیش‌پردازشی برای روش انتخاب ویژگی بسته‌بندی نیز به‌کار می‌رود [۲۰].

در این رویکرد زیرمجموعه‌ای از ویژگی‌ها بدون استفاده از هیچ الگوریتم یادگیری ماشینی، انتخاب می‌شود بنابراین بسیار سریع عمل می‌کند [۲۰]. اما عیب این روش‌ها این است که نمی‌توانند مشکل افزونگی ویژگی را برطرف کنند. افزونگی ویژگی بدان معناست که ویژگی غیرمؤثر بوده و باید کنار گذاشته شود ولی این روش تعداد ویژگی‌های افزونه را مشخص نمی‌کند و به همین دلیل باید تعداد ویژگی‌های مناسب تعیین شوند.

۲-۲- روش بسته‌بندی

به مجموعه روش‌هایی که از تابع ارزیابی مبتنی بر نرخ خطای روش یادگیری استفاده می‌کنند، روش‌های بسته‌بندی یا جعبه سیاه می‌گویند. در این روش هر زیرمجموعه جدید از ویژگی‌ها توسط تابع تولیدکننده ایجاد می‌شود که تولید این زیرمجموعه به استراتژی جستجو وابسته است. آنگاه زیرمجموعه تولیدشده به‌طور مکرر توسط روش یادگیری ماشین ارزیابی می‌شود. تعداد خطاهای مجموعه آزمون، یا همان نرخ خطای روش یادگیری، امتیاز زیرمجموعه را مشخص می‌کند [۲۱]. به‌طورکلی روش بسته‌بندی کارایی بهتری نسبت به روش فیلتر دارد، اما نسبت به روش فیلتر ازنظر محاسباتی پیچیده‌تر است.

در این پژوهش از پنج روش فیلتر شامل روش‌های امتیاز فیشر، شاخص جینی، کروسکال والیس، حداقل افزونگی-حداکثر همبستگی و آزمون T (این روش‌ها در [۱۸] شرح داده شده‌اند) و یک روش بسته‌بندی به‌نام FS برای ارائه روش انتخاب ویژگی ترکیبی استفاده شده است. علت استفاده از این روش‌ها آن است که این روش‌ها بهترین عملکرد را در پژوهش‌های اخیر در حوزه انتخاب ویژگی در پیش‌بینی اشکال نرم‌افزار [۱۱، ۱۵] داشته‌اند.

۳- کاربرد انتخاب ویژگی در پیش‌بینی اشکال نرم‌افزار

در طول سال‌های اخیر انتخاب ویژگی در کاربردهای بسیاری در حوزه مهندسی نرم‌افزار استفاده شده است [۱۰]. استفاده از این روش‌ها در حوزه پیش‌بینی کیفیت نرم‌افزار و خصوصاً پیش‌بینی اشکال نرم‌افزار نیز روبه‌رشد بوده است [۱۱]. چراکه در این حوزه با این چالش مواجه هستیم که بسیاری از ویژگی‌های استخراج‌شده از ماژول‌های نرم‌افزاری تأثیر منفی در دقت روش‌های یادگیری ماشین دارند و باید کنار گذاشته شوند. پژوهش‌های اخیری که بر تأثیر روش‌های انتخاب ویژگی در پیش‌بینی اشکال نرم‌افزار صورت پذیرفته است را می‌توان به سه دسته تقسیم کرد.

دسته اول پژوهش‌هایی هستند که ایده پیشنهادی آن‌ها تنها بر بررسی تأثیر روش‌های انتخاب ویژگی مختلف بر پیش‌بینی اشکال نرم‌افزار و مقایسه آن‌ها محدود شده است. برای نمونه در [۹] چهار روش مبتنی بر فیلتر و دو روش مبتنی بر بسته‌بندی به‌طور جداگانه و بدون هیچ ایده ترکیبی خاصی برای انتخاب ویژگی‌ها استفاده شده است. در پژوهش مذکور نشان داده شده است که می‌توان با استفاده از کمتر از ده درصد ویژگی‌های اولیه نیز به دقت بالا دست یافت. در [۲۲] نیز، تأثیر ترکیب روش‌های انتخاب ویژگی و یادگیری گروهی[۴] بر روی کارایی این روش‌ها در پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکال ارزیابی شده است. در این پژوهش ایده پیشنهادی، استفاده از روش یادگیری گروهی بوده است و برای بررسی تأثیر روش‌های انتخاب ویژگی بر کارایی روش یادگیری گروهی نیز از سه روش انتخاب روبه‌جلوی حریصانه (GFS[۵])، روش همبستگی پیرسون و روش امتیاز فیشر استفاده شده است.

دسته دوم پژوهش‌هایی هستند که به‌طور عمده روی جستجوی ویژگی‌های مرتبط با استفاده از ترکیب روش‌های انتخاب ویژگی تمرکز کرده‌اند. برای نمونه در [۱۵] ترکیبی از روش‌های انتخاب ویژگی برای پیش‌بینی اشکال ماژول‌های نرم‌افزاری سیستم مخابراتی بزرگی استفاده شده است. در این پژوهش روشی ترکیبی پیشنهاد شده است که در آن از هفت روش فیلتر و سه روش بسته‌بندی استفاده شده است. در روش ترکیبی ارائه‌شده، ابتدا تمامی ویژگی‌ها توسط روش‌های

فیلتر مرتب شده‌اند و سپس $\lceil \log_2 n \rceil$ (n تعداد کل ویژگی‌ها است) ویژگی اول از لیست مرتب‌شده ویژگی‌ها به‌عنوان مجموعه کل ویژگی‌ها به روش‌های بسته‌بندی داده شده است. نتایج این پژوهش نشان‌دهنده این است که حذف ۸۵ درصد از ویژگی‌ها تأثیر نامطلوبی روی نتایج نداشته، و در پاره‌ای از موارد نیز حتی موجب بهبود نتایج شده است. ایده پیشنهادی این پژوهش مشابه ایده پیشنهادی ما در روش انتخاب ویژگی دومرحله‌ای است، با این تفاوت که در روش پیشنهادی ما k ویژگی اول از لیست مرتب‌شده ویژگی‌ها (که توسط روش فیلتر ترکیبی مرتب می‌شود) به‌عنوان مجموعه ویژگی‌های اولیه در روش FS به‌کار می‌رود و مجموعه ویژگی‌های اضافه‌شونده نیز از مابقی ویژگی‌ها تشکیل می‌شود.

در [۲۳، ۲۴] نیز مطالعه تجربی کاملی بر ۱۷ روش یادگیری گروهی مختلف مبتنی بر روش‌های رتبه‌بندی ویژگی‌های نرم‌افزاری روی ۱۶ مجموعه از دادگان پیش‌بینی اشکال نرم‌افزار، صورت پذیرفته است. روش‌های رتبه‌بندی ویژگی‌ها به روش‌های انتخاب ویژگی مبتنی بر فیلتر اطلاق می‌شود. در این دو پژوهش ایده پیشنهادی روش یادگیری گروهی با محاسبه میانگین نتایج روش‌های فیلتر و انتخاب ویژگی‌های مرتبه بالا بوده است. روش پیشنهادی دو پژوهش فوق مشابه ایده پیشنهادی ما در مرحله اول از روش انتخاب ویژگی دومرحله‌ای (فیلتر ترکیبی وزن‌دار) است، با این تفاوت که در روش پیشنهادی ما به هر یک از روش‌های فیلتر نیز بر اساس میزان موفقیت آن‌ها در رتبه‌بندی ویژگی‌ها، وزنی نسبت داده شده است. به‌عبارت‌دیگر مزیت روش پیشنهادی ما نسبت به ایده پیشنهادی دو پژوهش فوق، اعمال تأثیر هر یک از روش‌های فیلتر بر رتبه هر یک از ویژگی‌های نرم‌افزاری با استفاده از وزن اختصاص‌یافته به آن روش فیلتر است.

دسته سوم پژوهش‌هایی هستند که به ترکیب روش‌های انتخاب ویژگی با دیگر روش‌های پرکاربرد و موردنیاز در پیش‌بینی اشکال نرم‌افزار پرداخته‌اند. این روش‌ها شامل روش‌های مقابله با مشکل عدم توازن کلاس [۲۵] و خوشه‌بندی دادگان [۱۶] هستند. برای نمونه در [۲۶] ترکیبی از روش‌های بهینه‌سازی فرا-ابتکاری[۶] (روش‌های الگوریتم ژنتیک (GA[۷]) و بهینه‌سازی ازدحام ذرات (PSO[۸])) و روش کیسه (Bag[۹]) برای بهبود دقت پیش‌بینی اشکال نرم‌افزار ارائه شده است. روش‌های بهینه‌سازی فرا-ابتکاری، برای انتخاب ویژگی و روش Bag برای مقابله با مشکل عدم توازن کلاس استفاده شده است. استفاده از روش‌های بهینه‌سازی فرا-ابتکاری موجب افزایش توانایی یافتن راه‌حل‌هایی با کیفیت بالا در بازه زمانی معقول می‌شود. هدف پژوهش مذکور از انتخاب روش‌های فوق برای انتخاب ویژگی بهبود دقت پیش‌بینی بوده است. به‌عبارت‌دیگر فاکتور سرعت از اهداف اصلی آن نبوده است، بنابراین از روش‌های انتخاب ویژگی فیلتر در فرآیند انتخاب ویژگی استفاده نشده است. اکثر پژوهش‌های حوزه پیش‌بینی اشکال، ایده پیشنهادی آن‌ها به انتخاب ویژگی و یا حل مشکل عدم توازن کلاس به‌صورت مجزا محدود شده است. اما در پژوهش مذکور این دو مبحث با یکدیگر ترکیب و به‌صورت روشی جامع ارائه شده‌اند.

در [۱۶] چارچوبی بنام خوشه‌بندی و رتبه‌بندی ویژگی‌ها برای پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکال پیشنهاد شده است. در چارچوب پیشنهادی آن‌ها برای انتخاب ویژگی از دو مبحث خوشه‌بندی ویژگی‌ها و رتبه‌بندی آن‌ها استفاده شده است. ایشان در روش پیشنهادی ابتدا ویژگی‌های اولیه را بر اساس معیار همبستگی به k خوشه تقسیم می‌کند. سپس مرتبط‌ترین ویژگی‌ها از هر خوشه را مبتنی بر میزان معیار وابستگی آن‌ها انتخاب می‌کند. همچنین در پژوهش خود از روش تقارن بدون قطعیت[۱۰] به‌عنوان معیار همبستگی استفاده کرده‌اند. از طرفی دیگر از روش‌های انتخاب ویژگی فیلتری همچون بهره‌وری اطلاعات (IG[۱۱]) و ReliefF به‌عنوان معیار وابستگی استفاده شده است. در این پژوهش‌ها مؤثرترین ویژگی‌های نرم‌افزاری جهت بهبود روند پیش‌بینی اشکال نرم‌افزار جستجو و معرفی شده‌اند. نتایج نهایی این پژوهش‌ها نشان‌دهنده تأثیر مثبت روش‌های انتخاب ویژگی در بهبود کارایی پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکال بوده است.

۴- روش انتخاب ویژگی پیشنهادی

در این پژوهش با توجه به دقت بالاتر روش‌های انتخاب ویژگی بسته‌بندی نسبت به فیلتر و در مقابل سرعت بالای انتخاب ویژگی‌ها توسط روش‌های فیلتر نسبت به بسته‌بندی، از ترکیب این دو روش برای انتخاب بهترین ویژگی‌های مؤثر در بهبود پیش‌بینی اشکال در ماژول‌های نرم‌افزاری استفاده شده است.

ایده پیشنهادی ما، استفاده از روش انتخاب ویژگی دومرحله‌ای برای انتخاب مؤثرترین ویژگی‌ها در فرآیند یادگیری است. روش پیشنهادی در مرحله اول با انتخاب ویژگی‌های اولیه توسط ترکیب وزن‌دار روش‌های فیلتر موجب بهبود سرعت در انتخاب ویژگی‌ها می‌شود. آنگاه در مرحله دوم با انتخاب ویژگی توسط روش‌های بسته‌بندی با توجه به ویژگی‌های انتخاب‌شده در مرحله اول، بهبود دقت به همراه افزایش سرعت حاصل می‌شود. در ادامه این دو مرحله بیان شده است. هر یک از متغیرهای استفاده‌شده در الگوریتم‌های روش پیشنهادی نیز در جدول ۱ معرفی شده‌اند.

مرحله اول: در روش‌های فیلتر مختلف با توجه به معیار در نظر گرفته‌شده توسط آن‌ها، به هر یک از ویژگی‌ها امتیازی داده می‌شود. امتیاز هر ویژگی در واقع وزن آن ویژگی است که مشخص‌کننده جایگاه ویژگی در لیست مرتب‌شده از ویژگی‌ها توسط روش فیلتر است. ایده پیشنهادی ما در مرحله اول ترکیب وزن‌دار روش‌های فیلتر است که مطابق الگوریتم ۱ به‌صورت مراحل زیر است:

۱. وزن‌دهی مجدد به ویژگی‌ها با توجه به اولویت هر ویژگی در لیست مرتب‌شده توسط هر یک از روش‌های فیلتر (اولویت بالاتر در لیست = مقدار وزن بیشتر- سطرهای ۱ تا ۱۰ الگوریتم ۱): در این مرحله

بر اساس لیست مرتب‌شده از ویژگی‌ها که روش فیلتر به‌عنوان خروجی برمی‌گرداند، وزن‌دهی مجدد انجام می‌شود و به ویژگی‌های مرتبط‌تر (و دارای اولویت بالاتر در لیست) وزن (w) بیشتری داده می‌شود.

۲. محاسبه وزن برای هر یک از روش‌های فیلتر بر اساس میزان موفقیت آن‌ها در اولویت‌بندی مؤثر ویژگی‌ها با استفاده از دادگان آموزش و اعتبارسنجی (سطرهای ۱۱ تا ۳۰ الگوریتم ۱): در این مرحله با هر یک از روش‌های فیلتر ویژگی‌های دادگان آموزش را اولویت‌بندی می‌کنیم و با بخش اعتبارسنجی میزان دقت هر یک از روش‌های فیلتر را ارزیابی می‌کنیم. آنگاه روش‌ها را بر اساس میزان موفقیتشان مرتب کرده و به روشی که کمترین دقت را دارد وزن ۰/۱ اختصاص می‌یابد و به دیگر روش‌ها بر اساس میزان اختلاف دقت آن‌ها نسبت به دقت روش قبلی وزنی معادل «میزان اختلاف تقسیم بر ۱۰ و به‌اضافه وزن روش قبلی» اختصاص می‌دهیم. برای نمونه به روشی که میزان دقت آن ۵ درصد بیشتر از روش با وزن ۰/۱ است، وزن ۰/۶ تعلق می‌گیرد.

۳. وزن‌دهی نهایی به ویژگی با ضرب وزن آن در وزن هر یک از روش‌های فیلتر و میانگین‌گیری از وزن جدید ویژگی با محاسبه حاصل جمع وزن‌های ویژگی در هر روش فیلتر و تقسیم آن به تعداد کل روش‌های فیلتر (سطرهای ۳۱ تا ۳۶ الگوریتم ۱): اگر ویژگی را با x نشان دهیم که دارای وزن اولیه W است و همچنین M روش فیلتر داشته باشیم که $fm_j.W$ بیانگر وزن روش فیلتر jام است. وزن نهایی با رابطه (۱) محاسبه می‌شود.

$$x.NewW = \frac{\sum_{j=1}^{M} fm_j.W \times x.W}{M} \quad (۱)$$

مرحله ۳ برای اعمال تأثیر میزان موفقیت روش‌های فیلتر در وزن‌دهی به ویژگی‌ها است. این روش به‌عنوان مرحله اول از روش انتخاب ویژگی دومرحله‌ای استفاده می‌شود به همین دلیل در سطرهای ۳۷ تا ۴۷ الگوریتم ۱ میزان k بهینه برای این روش محاسبه شده است، چراکه یکی از پارامترهای ورودی مرحله دوم است.

مرحله دوم: این مرحله با توجه به اولویت‌بندی ویژگی‌ها توسط روش فیلتر ترکیبی در مرحله اول آغاز می‌شود. در این مرحله مطابق الگوریتم ۲، از روش انتخاب ویژگی بسته‌بندی FS برای کاهش ویژگی‌ها و بهبود دقت پیش‌بینی اشکال در ماژول‌های نرم‌افزاری استفاده شده است. در روش FS، k ویژگی اول از روش فیلتر ترکیبی به‌عنوان مجموعه اولیه و مابقی ویژگی‌ها به‌عنوان مجموعه اضافه شونده در نظر گرفته شده است. بنابراین ما این روش را توسعه داده‌ایم.

مراحل انتخاب ویژگی پیشنهادی در مرحله دوم به‌صورت الگوریتم ۲ است. همچنین مراحل انتخاب ویژگی توسط روش FS معمولی نیز به‌صورت الگوریتم ۳ (بازنویسی مجدد الگوریتم ۱ از [۲۷]) است.

جدول ۱: متغیرهای استفاده‌شده در الگوریتم‌های روش پیشنهادی

نماد متغیر	شرح متغیر	نماد متغیر	شرح متغیر
X^{train}	مجموعه دادگان آموزشی	accuracy	میزان دقت
x^t	یک نمونه آموزشی	best	بهترین میزان دقت در هر تکرار الگوریتم انتخاب ویژگی
r^t	برچسب نمونه آموزشی (مستعد اشکال/ غیر مستعد اشکال)	$fm_i.bestAcc$	میزان دقت روش فیلتر iام
Nt	تعداد کل نمونه آموزشی	FilterList	لیست مرتب‌شده روش‌های فیلتر بر اساس میزان دقت آن‌ها
D	تعداد ویژگی‌های نمونه آموزشی	$fm_i.W$	وزن روش فیلتر iام
$X^{validate}$	مجموعه دادگان آزمون	d	میزان اختلاف دقت روش فیلتر iام با روش فیلتر i-1ام
x^v	یک نمونه آزمون	$x_i^t.NewW$	میزان وزن جدید ویژگی iام نمونه آموزشی با روش فیلتر ترکیبی وزن‌دار پیشنهادی
r^v	برچسب نمونه آزمون (مستعد اشکال/ غیر مستعد اشکال)	newFList	لیست ویژگی‌های مرتب‌شده توسط روش فیلتر ترکیبی وزن‌دار پیشنهادی
FM	مجموعه روش‌های فیلتر	optimalk	تعداد ویژگی‌های بهینه برای روش فیلتر ترکیبی وزن‌دار پیشنهادی
Nv	تعداد کل نمونه آزمون	S	لیست کل ویژگی‌های دادگان
fm_i	روش فیلتر iام	IS	لیست ویژگی‌های اولیه در روش FS
M	تعداد کل روش‌های فیلتر	AS	لیست ویژگی‌های اضافه‌شونده در روش FS
$FList_i$	لیست ویژگی‌های مرتب‌شده توسط روش فیلتر iام	FSFList	لیست ویژگی‌های نهایی انتخاب‌شده توسط روش FS
w	متغیری برای نگه‌داری موقت تعداد ویژگی‌ها	EM	میزان معیار ارزیابی (دقت)
fnum	شماره ویژگی	Max	متغیری برای نگه‌داری تعداد کل ویژگی‌های اضافه‌شونده در FS
$x_{fnum}^t.W$	وزن ویژگی با شماره fnum از نمونه آموزشی	s	ویژگی انتخاب‌شده از مجموعه ویژگی‌های اضافه‌شونده
train	متغیری برای نگه‌داری موقت نمونه‌های آموزشی با ویژگی‌های انتخاب‌شده توسط روش فیلتر	best-feature	متغیری برای نگه‌داری موقت بهترین ویژگی منتخب FS در هر تکرار الگوریتم
validate	متغیری برای نگه‌داری موقت نمونه‌های آزمون با ویژگی‌های انتخاب‌شده توسط روش فیلتر	n	متغیری برای نگه‌داری موقت تعداد کل ویژگی‌های باقی‌مانده از مجموعه ویژگی‌های اضافه‌شونده در روش FS

الگوریتم ۱: روش فیلتر ترکیبی وزن‌دار پیشنهادی (مرحله اول از روش انتخاب ویژگی دومرحله‌ای)

Fused weighted filter method (X^{train}, $X^{validate}$, FM)

Input:

$X^{train} = \{x^t, r^t\}_{t=1}^{Nt}$ // x^t is a train sample, $r^t \in \{nfp, fp\}$, Nt is the number of train samples, $x^t \in [x_1^t, x_2^t, x_3^t, ..., x_D^t]$ and D is the number of features.

$X^{validate} = \{x^v, r^v\}_{v=1}^{Nv}$ // x^v is a validate sample, $r^v \in \{nfp, fp\}$, Nv is the number of validate samples, $x^v \in [x_1^v, x_2^v, x_3^v, ..., x_D^v]$ and D is the number of features.

$FM = \{fm_i\}_{i=1}^{M}$ // fm is Filter method and M is the number of Filter methods.

Method:

```
for i=1: M
    FListi = fmi (xt);      //FListi is ordered List of features by fmi.
    w=D;
    // Assignment new weight to features for each of filter methods.
    for j=1: D
        fnum=FListi(j);      //fnum is feature's number.
        x^t_fnum.W = w;
        w=w-1;
    end
end
// calculate weight for each of filter methods.
for i=1: M
    FList = fmi (xt);      // FList is ordered List of features by fmi.
    for j=1: size(FList)
        train = train + FList.x^t_j;
        validate = validate + FList.x^v_j;
        accuracy = classifier (train, validate);
        if accuracy>best result in this iteration
            best=accuracy;
        end
    end
    fmi.bestAcc= best;
end
FilterList= sort (fm.bestAcc, 'ascend');  // FilterList is ordered list of Filter methods.
i=FilterList(1);
fmi.W=0.1;
for i=2: M
    d= fmi.bestAcc – fmi-1.bestAcc;
    fmi.W= fmi-1.W + (d/10);
end
// Assignment new weight to features by WF method.
for i=1: D
    x^t_i.NewW = (sum_{j=1}^{M} fmj.W × x^t_i.W) / M;
    newFList.Add ( x^t_i );
end
[newFList] = sort (newFList.NewW, 'descend');
// calculate optimal k for feature selection by WF method.
for j=1: size(newFList)
    train = train + newFList.x^t_j;
    validate = validate + newFList.x^v_j;
    accuracy = classifier (train, validate);
    if accuracy>best result in this iteration
        best=accuracy;
        k=j;
    end
end
optimalk= k;
```

Output:

newFList // the ordered subset of the features by WF method.

optimalk // the optimal k for feature selection by WF method.

الگوریتم ۲: روش انتخاب روبه‌جلو توسعه‌یافته پیشنهادی (مرحله دوم از روش انتخاب ویژگی دومرحله‌ای)

Enhanced forward feature selection method (X, FM) در دو پاراگراف آخر از صفحه ششم مقاله ۲ بار به این الگوریتم اشاره شده است.

Input:

X=$\{x^t, r^t\}_{t=1}^{N}$ // x^t is a train sample, $r^t \in$ {nfp, fp}, N is the number of samples, $x^t \in [x_1^t, x_2^t, x_3^t, \dots, x_D^t]$, and D is the number of features.

FM=$\{fm_i\}_{i=1}^{M}$ // fm_i is a known Filter method and M is the number of Filter methods.

Method:

2: [X^{train} $X^{validate}$]=10-fold cross validation(X);
3: [FList k]=Fused weighted filter method (X^{train} , $X^{validate}$, FM);
4: // Enhanced forward feature selection method:
4: S= FList; //FList is $[x_1^t, x_2^t, x_3^t, \dots, x_D^t]$.
5: IS= $[x_1^t, x_2^t, x_3^t, \dots, x_k^t]$; //IS is initial subset for enhanced forward feature selection method.
6: AS=$[x_{k+1}^t, x_{k+2}^t, x_{k+3}^t, \dots, x_D^t]$; //AS is S-IS and it is additive subset for enhanced forward feature selection method.
7: [FSFList, EM] = Forward selection method (X^{train} , $X^{validate}$, IS, AS)

Output:

FSFList // the optimal subset of the features by forward selection method.
EM // the value of evaluation measures by forward selection method.

الگوریتم ۳: روش FS

Forward selection method (X_t^{rain} , $X^{validate}$, IS, AS)

Input:

X^{train} =$\{x^t, r^t\}_{t=1}^{Nt}$ // x^t is a train sample, $r^t \in$ {nfp, fp}, Nt is the number of train samples, $x^t \in [x_1^t, x_2^t, x_3^t, \dots, x_D^t]$, and *D* is the number of features.

$X^{validate}$ =$\{x^v, r^v\}_{v=1}^{Nv}$ // x^v is a validate sample, $r^v \in$ {nfp, fp}, Nv is the number of validate samples, $x^v \in [x_1^v, x_2^v, x_3^v, \dots, x_D^v]$, and D is the number of features.

IS // initial subset for forward feature selection method.
AS // additive subset for forward feature selection method.

Method:

```
1:  n=1;
2:  Max = Size(AS);
3:  while (n ≤ Max)
4:      while(s = selected next element of AS)
5:          S=IS ∪ s;
6:          [accuracy] = Classifier (X^train , X^validate , S);
7:          If accuracy > Best result in this iteration
8:              Best = accuracy;
9:              best-feature=s;
10:         end
11:     end
12:     if Best > previous Best result
13:         IS=IS ∪ best-feature;
14:         AS=AS – best-feature;
15:         EM = Best;
16:     else
17:         break;
18:     end
19:     n=n+1;
20: end
```

Output:

IS // the optimal subset of the features
EM // the best value of evaluation measures by features in IS

۵- دادگان‌های ارزیابی مدل پیشنهادی

جهت بررسی مؤثر بودن روش‌های پیشنهادی در پیش‌بینی اشکال نرم‌افزار، از ۸ دادگان مخصوص پیش‌بینی اشکال نرم‌افزار استفاده شده است. در این دادگان‌ها، آخرین ویژگی برچسب کلاس هر نمونه است. در این ویژگی، مقدار صفر نشان‌دهنده غیرمستعد اشکال بودن ماژول و مقادیر غیر صفر نشان‌دهنده مستعد اشکال بودن آن است. جدول ۲ اطلاعات این دادگان را نشان می‌دهد. تعداد نمونه‌ها نشان‌دهنده تعداد ماژول‌های نرم‌افزاری است. هر نمونه بیانگر ماژولی است که ویژگی‌های مختلفی از آن استخراج شده است. ویژگی‌های هر یک از دادگان نیز در جدول مذکور با علامت × نشان داده شده است [۲۸].

جدول ۲: اطلاعات دادگان پیش‌بینی اشکال نرم‌افزار

نماد ویژگی	نام کامل ویژگی	معادل فارسی	CM1	KC3	MC1	MC2	MW1	PC1	PC2	PC3
x_1	Line count of code	تعداد کل خطوط کد	×	×	×	×	×	×	×	×
x_2	McCable's cyclomatic complexity	پیچیدگی نمودار جریان	×	×	×	×	×	×	×	×
x_3	McCable's essential complexity	پیچیدگی تجزیه	×	×	×	×	×	×	×	×
x_4	McCable's design complexity	پیچیدگی طراحی	×	×	×	×	×	×	×	×
x_5	Total number of operators	تعداد کل عملگرها	×	×	×	×	×	×	×	×
x_6	Total number of operands	تعداد کل عملوندها	×	×	×	×	×	×	×	×
x_7	Number of unique operators	تعداد کل عملگرهای یکتا	×	×	×	×	×	×	×	×
x_8	Number of unique operands	تعداد کل عملوندهای یکتا	×	×	×	×	×	×	×	×
x_9	Number of unique operators and operands	تعداد کل عملگرها و عملوندهای یکتا	×	×	×	×	×	×	×	×
x_{10}	Halstead's volume	حجم	×	×	×	×	×	×	×	×
x_{11}	Halstead's difficult	دشواری	×	×	×	×	×	×	×	×
x_{12}	Halstead's length	طول ماژول	×	×	×	×	×	×	×	×
x_{13}	Halstead's content	محتوا	×	×	×	×	×	×	×	×
x_{14}	Halstead's effort	میزان تلاش	×	×	×	×	×	×	×	×
x_{15}	Halstead's error estimate	میزان برآورد خطا	×	×	×	×	×	×	×	×
x_{16}	Halstead's programing time	میزان زمان برنامه‌نویسی	×	×	×	×	×	×	×	×
x_{17}	Number of blank lines	تعداد خطوط خالی ماژول	×	×	×	×	×	×	×	×
x_{18}	Number of comment-only lines	تعداد خطوط توضیحات از ماژول	×	×	×	×	×	×	×	×
x_{19}	Number of code-only lines	تعداد خطوطی که تنها شامل کد هستند	×					×		
x_{20}	Number of lines with both code and comments	تعداد خطوط توضیحات و کد	×	×	×	×	×	×	×	×
x_{21}	Number of branches	تعداد کل نمودارهای جریان	×	×	×	×	×	×	×	×
x_{22}	Number of condition	تعداد دستورهای شرطی		×	×	×	×		×	×
x_{23}	Call pairs	فراخوانی دوجزئی		×	×	×	×		×	×
x_{24}	Cyclomatic density	تراکم مسیرهای نمودار جریان		×	×	×	×		×	×
x_{25}	Number of decision	تعداد دستورهای تصمیم‌گیری		×	×	×	×		×	×
x_{26}	Decision density	تراکم مسیرهای دستورهای تصمیم‌گیری		×	×	×	×		×	×
x_{27}	Design density	تراکم طراحی		×	×	×	×		×	×
x_{28}	Number of edge	تعداد لبه‌ها یا مرزها		×	×	×	×		×	×
x_{29}	Essential density	تراکم ضروری		×	×	×	×		×	×
x_{30}	LOC executable	تعداد خط‌های قابل‌اجرا		×	×	×	×		×	×
x_{31}	Number of parameter	تعداد پارامترهای ورودی		×	×	×	×		×	×
x_{32}	Global data complexity	پیچیدگی داده‌های سراسری		×	×	×				
x_{33}	Global data density	تراکم داده‌های سراسری		×	×	×				
x_{34}	Halstead's level	سطح		×	×	×	×		×	×
x_{35}	Maintenance severity	شدت تعمیر و نگه‌داری		×	×	×	×		×	×
x_{36}	Number of modified condition	تعداد شرط‌های اصلاح‌شده		×	×	×	×		×	×
x_{37}	Number of multiple condition	تعداد شرط‌های چندگانه		×	×	×	×		×	×
x_{38}	Number of node	تعداد گره‌ها		×	×	×	×		×	×
x_{39}	Normalized cyclomatic complexity	میزان نرمال‌شده تعداد مسیرهای مستقل خطی در نمودار جریان ماژول		×	×	×	×		×	×
x_{40}	Percent comments	میزان درصد توضیحات در کد		×	×	×	×		×	×
		تعداد ویژگی‌ها	۲۱	۳۹	۳۹	۳۹	۳۷	۲۱	۳۷	۳۷
		تعداد نمونه‌ها	۴۹۸	۴۵۸	۸۹۶	۱۶۱	۴۰۳	۱۱۰۹	۷۷۸	۷۵۳

۶- معیارهای ارزیابی مدل پیشنهادی

ازجمله عوامل مؤثر در مقایسه مدل با دیگر مدل‌ها، سنجش آن با معیار ارزیابی مناسب است. در این پژوهش جهت ارزیابی دقت طبقه‌بندها در تشخیص ماژول‌های نرم‌افزاری مستعد اشکال با توجه به معیارهای استفاده‌شده در پژوهش‌هایی که با نتایج روش‌های پیشنهادی ما مورد مقایسه قرار گرفته‌اند، از معیار سطح زیر منحنی مشخصه عملکرد سیستم (AUC[۱۲]) استفاده شده است. معیار AUC به‌طور گسترده برای اندازه‌گیری میزان کارایی مدل طبقه‌بندی استفاده می‌شود. این معیار را می‌توان به‌عنوان یک توصیف آماری استفاده کرد، تا تخمین زده شود که احتمال این‌که یک مدل پیش‌بینی، یک نمونه دارای اشکال را بیشتر از یک نمونه فاقد اشکال، تشخیص دهد، چه مقدار است [۲۹].

در این معیار، از چهار معیار شامل معیارهای مثبت واقعی (TP[۱۳])، مثبت کاذب (FP[۱۴])، منفی واقعی (TN[۱۵])، منفی کاذب (FN[۱۶])، نرخ مثبت واقعی (TP_r[۱۷]) و نرخ مثبت کاذب (FP_r[۱۸]) استفاده شده است.

TP بیانگر تعدادِ ماژول‌های دارای اشکال که به‌درستی مستعد اشکال پیش‌بینی شده‌اند، است. FP بیانگر تعداد ماژول‌های فاقد اشکال است که به‌اشتباه مستعد اشکال پیش‌بینی شده‌اند. TN بیانگر تعداد ماژول‌های فاقد اشکال که به‌درستی غیرمستعد اشکال پیش‌بینی شده‌اند، است. FN نیز تعداد ماژول‌های دارای اشکال که به‌اشتباه غیرمستعد اشکال پیش‌بینی شده‌اند، است. معیار TP_r بیانگر نرخ تشخیص صحیح دسته مثبت توسط طبقه‌بند است. معیار FP_r نرخ تشخیص اشتباه دسته منفی را بیان می‌کند [۲۹]. دو معیار TP_r و FP_r به‌ترتیب با رابطه‌های (۲) و (۳) محاسبه می‌شوند. منظور از دسته مثبت، نمونه‌های دارای برچسب کلاس غیرصفر هستند که همان ماژول‌های مستعد اشکال می‌باشند. همچنین منظور از دسته منفی نمونه‌های دارای برچسب کلاس صفر یعنی ماژول‌های غیر مستعد اشکال هستند. مقدار معیار AUC با رابطه (۴) محاسبه می‌شود [۳۰].

$$TP_r = \frac{TP}{TP + FN} \quad (۲)$$

$$FP_r = \frac{FP}{FP + TN} \quad (۳)$$

$$AUC = \frac{1 + TP_r - FP_r}{2} \quad (۴)$$

۷- نتایج تجربی آزمایش‌ها

تمامی دادگان در این پژوهش ابتدا به دو بخش دادگان آموزش و دادگان آزمون تقسیم شده‌اند. تقسیم‌بندی به‌صورت، ۹۰ درصد برای آموزش و ۱۰ درصد برای آزمون (ارزیابی A1) و ۱۰ درصد برای آموزش و ۹۰ درصد برای آزمون (ارزیابی A2) انجام شده است. آنگاه بخش آموزش هم، خود به دو بخش آموزش و اعتبارسنجی تقسیم شده است که برای تعیین پارامترهای مدل پیشنهادی مورداستفاده قرار گرفته است. آنچه مسلم است این است که نتیجه حاصل از ارزیابی دادگانی که در آموزش استفاده شده‌اند به‌عنوان نتیجه ارزیابی کلی مدنظر قرار نمی‌گیرد، بلکه آنچه به‌عنوان نتیجه ارزیابی کارایی روش مدنظر است، دقت آن روش روی پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکالی که در بخش دادگان آزمون قرار دارند، است. برای این تقسیم‌بندی بخش آموزش به دو بخش آموزش و اعتبارسنجی از روش اعتبارسنجی متقابل k بخشی[۱۹] با k=10 استفاده شده است. همچنین برای طبقه‌بندی ماژول‌های نرم‌افزاری به دو دسته مستعد اشکال و غیرمستعد اشکال در این پژوهش از روش ماشین بردار پشتیبان دوقلو هموارشده با حاشیه پارامتریک (STPMSVM[۲۰]) [۱۷] استفاده شده است. علت استفاده از روش مذکور به‌عنوان طبقه‌بند، برتری این روش نسبت به دیگر طبقه‌بندها در پیش‌بینی اشکال نرم‌افزار در پژوهش [۳۱] است.

مرحله دوم از روش انتخاب ویژگی دومرحله‌ای، با ویژگی‌های به‌دست‌آمده توسط روش فیلتر ترکیبی وزن‌دار در مرحله اول، آغاز می‌شود. در این مرحله از روش‌های انتخاب ویژگی بسته‌بندی برای کاهش ویژگی‌ها و بهبود دقت پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکال استفاده شده است. روش بسته‌بندی استفاده‌شده در این مرحله روش FS است که پارامترهای ورودی آن مطابق الگوریتم ۲ اصلاح شده است و اصطلاحاً توسعه داده شده است.

برای روش فیلتر ترکیبی پیشنهادشده، میانگین تعداد ویژگی‌های انتخاب‌شده از هر یک از دادگان حدود ۵۶ درصد است. بنابراین حدود ۵۶ درصد از ویژگی‌ها (که برترین ویژگی‌ها ازنظر روش فیلتر ترکیبی وزن‌دار پیشنهادی هستند) ثابت در نظر گرفته می‌شود و ۴۴ درصد از ویژگی‌های باقی‌مانده توسط روش FS موردبررسی قرار می‌گیرد که این مسئله علاوه‌بر بهبود دقت، موجب نصف شدن زمان اجرای روش FS می‌شود. بنابراین می‌توان نتیجه گرفت که در پروژه‌های نرم‌افزاری واقعی که تعداد ویژگی‌ها و ماژول‌های نرم‌افزاری بسیار بیشتر از دادگان موردآزمایش قرارگرفته در این پژوهش هستند، الگوریتم پیشنهادی می‌تواند با انتخاب ویژگی‌های برتر، علاوه‌بر بهبود دقت پیش‌بینی، موجب کاهش زمان و حدوداً دو برابر شدن سرعت نیز شود.

در آزمایش روش پیشنهادی علاوه‌بر ارزیابی دقت، اقدام به اندازه‌گیری زمان اجرای الگوریتم پیشنهادی (شامل روش فیلتر ترکیبی وزن‌دار و روش FS توسعه‌داده‌شده) و روش FS معمولی نموده‌ایم. اندازه‌گیری زمان با استفاده از توابع Tic و Toc نرم‌افزار متلب انجام شده است. تمامی الگوریتم‌های استفاده‌شده در روش پیشنهادی در محیط متلب و روی یک کامپیوتر شخصی با پردازنده چهار هسته‌ای، RAM معادل ۴ گیگابایت و با ویندوز ۷ پیاده‌سازی شده است.

نتایج نهایی روش انتخاب ویژگی ترکیبی دومرحله‌ای و همچنین نتایج روش FS معمولی و نتایج عدم استفاده از این روش (استفاده از تمام ویژگی‌ها) در جدول‌های ۳ و ۴ به‌صورت دقت و زمان اجرا (برحسب ثانیه) نشان داده شده است. میزان بهبود دقت روش

پیشنهادی نسبت به استفاده از تمام ویژگی‌ها و میزان افزایش سرعت روش پیشنهادی نسبت به روش FS معمولی نیز در جدول‌های مذکور نشان داده شده است.

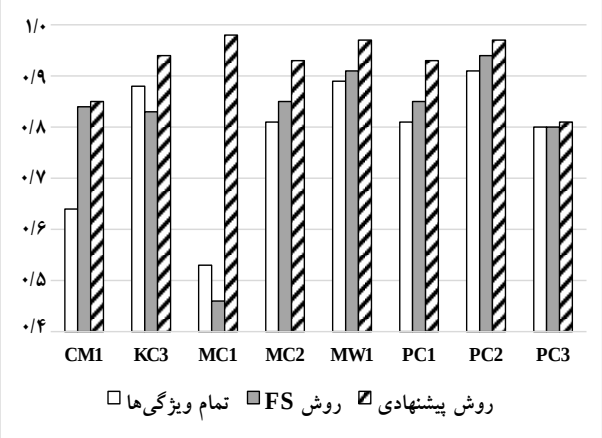
جدول ۳: نتایج روش انتخاب ویژگی پیشنهادی در مقایسه با نتایج روش FS معمولی و استفاده از تمام ویژگی‌ها با ارزیابی A1

دادگان	تمام ویژگی‌ها	روش FS معمولی		روش پیشنهادی		میزان بهبود	میزان افزایش
		دقت	زمان	دقت	زمان	دقت	سرعت
CM1	۰/۶۴	۰/۸۴	۱/۱۱	۰/۸۵	۰/۵۲	+۰/۲۱	+۰/۵۹
KC3	۰/۸۸	۰/۸۳	۰/۷۸	۰/۹۴	۰/۲۴	+۰/۰۶	+۰/۵۴
MC1	۰/۵۳	۰/۴۶	۱/۹۷	۰/۹۸	۱/۴۷	+۰/۴۵	+۰/۵
MC2	۰/۸۱	۰/۸۵	۰/۸۶	۰/۹۳	۰/۱۹	+۰/۱۲	+۰/۶۷
MW1	۰/۸۹	۰/۹۱	۰/۶۹	۰/۹۷	۰/۲۰	+۰/۰۸	+۰/۴۹
PC1	۰/۸۱	۰/۸۵	۱/۳۷	۰/۹۳	۰/۸۶	+۰/۱۲	+۰/۵۱
PC2	۰/۹۱	۰/۹۴	۱/۵۴	۰/۹۷	۰/۴۶	+۰/۰۶	+۱/۰۸
PC3	۰/۸۰	۰/۸۰	۲/۰۶	۰/۸۱	۰/۳۷	+۰/۰۱	+ ۱/۶۹
میانگین	۰/۷۸	۰/۸۱	۱/۳۰	۰/۹۲	۰/۵۴	+۰/۱۴	+ ۰/۷۶

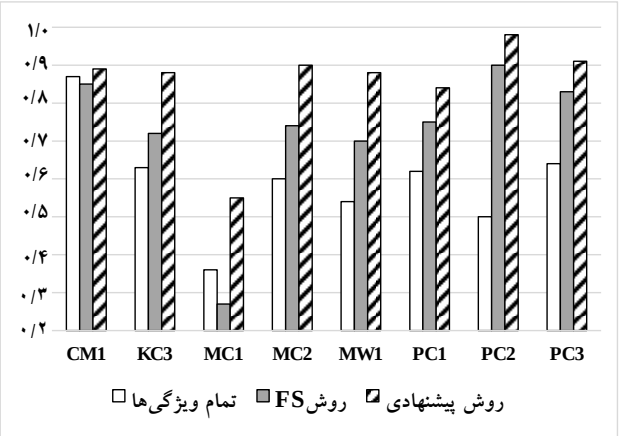
جدول ۴: نتایج روش انتخاب ویژگی پیشنهادی در مقایسه با نتایج روش FS معمولی و استفاده از تمام ویژگی‌ها با ارزیابی A2

دادگان	تمام ویژگی‌ها	روش FS معمولی		روش پیشنهادی		میزان بهبود	میزان افزایش
		دقت	زمان	دقت	زمان	دقت	سرعت
CM1	۰/۸۷	۰/۸۵	۱/۴۱	۰/۸۹	۰/۵۱	+۰/۰۴	+۰/۹
KC3	۰/۶۳	۰/۷۲	۰/۷۸	۰/۸۸	۰/۰۶	+۰/۱۶	+۰/۷۲
MC1	۰/۳۶	۰/۲۷	۰/۹۴	۰/۵۵	۰/۶۱	+۰/۲۸	+۰/۳۳
MC2	۰/۶۰	۰/۷۴	۰/۷۵	۰/۹۰	۰/۱۰	+۰/۱۶	+۰/۶۵
MW1	۰/۵۴	۰/۷۰	۰/۶۰	۰/۸۸	۰/۲۲	+۰/۱۸	+۰/۳۸
PC1	۰/۶۲	۰/۷۵	۰/۸۷	۰/۸۴	۰/۸۴	+۰/۱۲	+۰/۰۳
PC2	۰/۵۰	۰/۹۰	۰/۲۰	۰/۹۸	۰/۰۹	+۰/۰۸	+۰/۱۱
PC3	۰/۶۴	۰/۸۳	۰/۷۵	۰/۹۱	۰/۱۴	+۰/۰۸	+ ۰/۶۱
میانگین	۰/۵۹	۰/۷۲	۰/۷۰	۰/۸۵	۰/۳۲	+۰/۱۴	+ ۰/۴۷

مطابق جدول ۳ مشاهده می‌شود که در تمامی دادگان میزان دقت بهبود یافته است. برای نمونه روش پیشنهادی در دادگان CM1 با میزان دقت، ۰/۸۵ ماژول‌های نرم‌افزاری مستعد اشکال را به‌درستی پیش‌بینی کرده است که بیشتر از میزان دقت به‌دست‌آمده با تمام ویژگی‌ها (۰/۶۴) است. نکته قابل‌توجه آن است که روش پیشنهادی به‌طور میانگین ۱۴ درصد بهبود دقت داشته است. همچنین این نکته نیز قابل‌توجه است که در روش پیشنهادی حدود ۵۶ درصد از ویژگی‌ها در مرحله اول توسط روش فیلتر انتخاب شده‌اند و مابقی ویژگی‌ها توسط روش FS موردبررسی قرار گرفته‌اند، بنابراین می‌توان نتیجه گرفت که با کاهش ۵۰ درصدی پیچیدگی زمانی نسبت به، استفاده از روش FS معمولی، روش پیشنهادی سرعت اجرای حدوداً دو برابر داشته است. مطابق جدول ۳ مشاهده می‌شود که روش پیشنهادی نسبت به روش FS معمولی به‌طور میانگین ۰/۷۶ ثانیه در ارزیابی A1 سریع‌تر عمل کرده است. همچنین مطابق جدول ۴ مشاهده می‌شود که روش پیشنهادی در ارزیابی A2 نسبت به روش FS معمولی به‌طور میانگین ۰/۴۷ ثانیه سریع‌تر عمل کرده است. منحنی میزان AUC روش پیشنهادی بر ۸ دادگان ارزیابی ناسا برای ارزیابی‌های A1 و A2 به‌ترتیب در نمودارهای شکل ۱ و شکل ۲ نشان داده شده است.



شکل ۱: نمودار منحنی میزان AUC روش پیشنهادی در مقایسه با روش FS و استفاده از تمام ویژگی‌ها با ارزیابی A1



شکل ۲: نمودار منحنی میزان AUC روش پیشنهادی در مقایسه با روش FS و استفاده از تمام ویژگی‌ها با ارزیابی A2

با توجه به نمودارهای شکل ۱ و شکل ۲ مشاهده می‌شود که روش پیشنهادی میزان AUC قابل قبولی برای پیش‌بینی اشکال در این دادگان داشته است و در تمامی دادگان میزان دقت با این روش بهبود یافته است.

بررسی ویژگی‌های انتخاب‌شده از دادگان توسط روش پیشنهادی بیانگر آن است که پنج ویژگی بهترین و مؤثرترین ویژگی‌های نرم‌افزاری برای بهبود پیش‌بینی اشکال نرم‌افزار با روش انتخاب ویژگی پیشنهادی به‌طور مشترک در هر دو ارزیابی A1 و A2 هستند. این ویژگی‌ها در جدول ۵ آورده شده‌اند. همچنین با بررسی دقیق‌تر نتایج مشاهده می‌شود که سه ویژگی نیز غیرمؤثرترین ویژگی‌های نرم‌افزاری در پیش‌بینی اشکال نرم‌افزار با روش انتخاب ویژگی پیشنهادی به‌طور

مشترک در هر دو ارزیابی A1 و A2 هستند. این ویژگی‌ها نیز در جدول ۶ آورده شده‌اند.

نتایج حاصل از روش انتخاب ویژگی پیشنهادی در جدول ۷، با نتایج پنج پژوهش دیگر مقایسه شده است. از هر یک از این پژوهش‌ها، نتایج روش‌هایی که بهترین عملکرد را داشته‌اند، بیان شده است. در این جدول بهترین نتایج برای هر یک از دادگان برجسته شده است. ما در این پژوهش از ۸ دادگان استفاده کرده‌ایم، اما با توجه به این‌که در اکثر پژوهش‌های مرتبط تنها از تعداد اندکی از دادگان استفاده شده است، برای برخی از دادگان توسط دیگر پژوهش‌ها نتایجی اعلام نشده است، بنابراین خانه مربوطه برای این پژوهش‌ها در جدول با خط تیره مشخص شده است. نتایج مقایسه‌ها نشان‌دهنده این است که روش پیشنهادی این پژوهش در مقایسه با پژوهش‌های مشابه، در اکثر دادگان عملکرد بهتری در پیش‌بینی ماژول‌های نرم‌افزاری مستعد اشکال نسبت به پژوهش‌های مشابه داشته است.

جدول ۵: ویژگی‌های برگزیده توسط روش پیشنهادی

نماد ویژگی	نام کامل ویژگی	معادل فارسی
x_1	Line count of code	تعداد کل خطوط کد
x_9	Number of unique operators and operands	تعداد کل عملگرها و عملوندهای یکتا
x_{18}	Number of comment-only lines	تعداد خطوط توضیحات از ماژول
x_{31}	Number of parameter	تعداد پارامترهای ورودی
x_{36}	Number of modified condition	تعداد شرط‌های اصلاح‌شده

جدول ۶: ویژگی‌های غیرمؤثر شناخته‌شده توسط روش پیشنهادی

نماد ویژگی	نام کامل ویژگی	معادل فارسی
x_{17}	Number of blank lines	تعداد خطوط خالی ماژول
x_{29}	Essential density	تراکم ضروری
x_{39}	Normalized cyclomatic complexity	میزان نرمال‌شده تعداد مسیرهای مستقل خطی در نمودار جریان ماژول

جدول ۷: مقایسه نتایج روش پیشنهادی با نتایج دیگر پژوهش‌ها

دادگان		CM1	KC3	MC1	MW1	MC2	PC1	PC2	PC3
این مقاله	A1	۰/۸۵	۰/۹۴	۰/۹۸	۰/۹۷	۰/۹۳	۰/۹۳	۰/۹۷	۰/۸۱
	A2	۰/۸۹	۰/۸۸	۰/۵۵	۰/۸۸	۰/۹۰	۰/۸۴	۰/۹۸	۰/۹۱
[۱۶]		۰/۷۸	۰/۸۲	-	۰/۷۹	۰/۷۱	۰/۷۹	-	۰/۸۰
[۲۲]		-	۰/۸۶	۰/۹۸	-	-	-	۰/۹۵	-
[۲۶]		۰/۷۶	۰/۷۹	-	۰/۷۵	۰/۷۸	۰/۸۵	۰/۹۲	۰/۷۹
[۳۲]		-	-	۰/۹۰	-	-	۰/۸۳	۰/۸۰	۰/۸۴
[۳۳]		۰/۷۹	۰/۸۳	۰/۹۷	۰/۷۸	۰/۷۷	۰/۸۹	۰/۹۰	۰/۸۵

۸- نتیجه‌گیری و پژوهش‌های پیش رو

در این پژوهش به‌دلیل اهمیت و کارایی روش‌های انتخاب ویژگی در بهبود عملکرد روش‌های یادگیری ماشین، روش‌های ترکیبی انتخاب ویژگی برای انتخاب مؤثرترین ویژگی‌های ماژول‌های نرم‌افزاری ارائه شدند. روش بسته‌بندی به‌علت استفاده از یک الگوریتم یادگیری ماشین دارای سرعت پایینی است اما این روش دقت بالایی دارد. در مقابل روش فیلتر سرعت عمل بالایی دارد ولی دقت چندان مناسبی را ارائه نمی‌دهد. با ترکیب نقاط قوت این دو روش یعنی عملکرد خوب روش انتخاب ویژگی بسته‌بندی و همچنین سرعت بالای روش انتخاب ویژگی فیلتر، ما به روش ترکیبی مناسبی برای انتخاب ویژگی‌ها دست یافته‌ایم.

در این پژوهش ما روش FS که یکی از بهترین روش‌های انتخاب ویژگی بسته‌بندی است را توسعه داده‌ایم. روش FS برای شروع به کار خود از یک مجموعه خالی به‌عنوان مجموعه ویژگی‌های اولیه استفاده می‌کند. ما در این پژوهش این مجموعه اولیه را با ویژگی‌های به‌دست‌آمده از ترکیب روش‌های فیلتر اصلاح کرده‌ایم. سپس این روش را روی دادگان پیش‌بینی اشکال نرم‌افزار ناسا مورد ارزیابی قرار دادیم. نتایج حاکی از مؤثر بودن روش مورداستفاده می‌باشند. بنابراین روش پیشنهادی ما توانسته است، بهترین ویژگی‌ها برای بهبود دقت پیش‌بینی اشکال را با سرعت بیشتری بیابد. در این پژوهش ما از پنج روش فیلتر و یک روش بسته‌بندی استفاده کردیم. ما قصد داریم در آینده روی ترکیبات دیگری از روش‌های فیلتر و بسته‌بندی کار کنیم.

مراجع

[1] I. Gondra, "Applying machine learning to software fault-proneness prediction," *J. Syst. Softw.*, vol. 81, no. 2, pp. 186-195, 2008.

[۲] علیقارداشی و زارع چاهوکی، «افزایش دقت در پیش‌بینی اشکال نرم‌افزار با استفاده از ماشین بردار پشتیبان دوقلو با طرح حداقل مربعات بازگشتی،» *هشتمین کنفرانس داده‌کاوی ایران*، تهران، ۱۳۹۳.

[3] E. Rashid, S. Patnaik, and A. Usmani, "Machine learning and its application in software fault prediction with similarity measures," *Comput. Vis. Robot.*, Springer India, vol. 332, pp. 37-45, 2015.

[4] C. Catal, "Software fault prediction: A literature review and current trends," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4626-4636, 2011.

[5] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *Softw. Eng. IEEE Trans.*, vol. 33, no. 1, pp. 2-13, 2007.

[6] G. Chandrashekar, and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16-28, 2014.

[۷] مالکی، *پیش‌بینی خطای نرم‌افزار بر اساس وابستگی‌های قطعات کد با استفاده از داده‌کاوی*، پایان‌نامه کارشناسی ارشد، دانشکده آموزش‌های الکترونیکی، دانشگاه شیراز، شیراز، ایران، ۱۳۹۰.

[8] E. Alpaydin, *Introduction to Machine Learning*, MIT press, 2014.

[9] S. Shivaji, S. Member, E. J. W. Jr, and S. Member, "Reducing features to improve code change-based bug prediction," *IEEE Trans. Softw. Eng.*, vol. 39, no. 4, pp. 552-569, 2013.

[10] S. Srivastava, N. Joshi, and M. Gaur, "A review paper on feature selection methodologies and their applications," *Int. J. Comput. Sci. Netw. Secur.*, vol. 14, no. 5, p. 78, 2014.
[11] K. Muthukumaran, A. Rallapalli, and N. L. B. Murthy, "Impact of feature selection techniques on bug prediction models," *Proc. 8th India Softw. Eng. Conf.*, pp. 120-129, 2015.
[12] J. Apolloni, G. Leguizamón, and E. Alba, "Two hybrid wrapper-filter feature selection algorithms applied to high-dimensional microarray experiments," *Applied Soft Computing*, vol. 38, pp. 922-932, 2016.
[13] A. Idris, and A. Khan, "Churn prediction system for telecom using filter–wrapper and ensemble classification," *The Computer Journal*, bxv123, 2016.
[14] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Future Generation Computer Systems*, vol. 55, pp. 376-390, 2016.
[15] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction : an investigation on feature selection techniques," *Softw. Pract. Exp.*, vol. 41, no. 5, pp. 579-606, 2011.
[16] S. Liu, X. Chen, W. Liu, J. Chen, Q. Gu, and D. Chen, "FECAR : a feature selection framework for software defect prediction," *Comput. Softw. Appl. Conf. (COMPSAC), IEEE 38th Annu*, pp. 426-435, 2014.
[17] Z. Wang, Y. Shao, and T. Wu, "A GA-based model selection for smooth twin parametric-margin support vector machine," *Pattern Recognit.*, vol. 46, no. 8, pp. 2267-2277, 2013.
[18] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, "Advancing feature selection research," *ASU Featur. Sel. Repos.*, pp. 1-28, 2010.
[19] A. Jović, and N. Bogunović, "A review of feature selection methods with applications," *Info. Comm. Tech., Elec. Microelec. (MIPRO), 38th International Convention on*, 2015.
[20] H. H. Hsu, C. W. Hsieh, and M. D. Lu, "Hybrid feature selection by combining filters and wrappers," *Expert Syst. Appl.*, vol. 38, no. 7, pp. 8144-8150, 2011.
[21] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: a review," *Data Classif. Algorithms Appl.*, p. 37, 2014.
[22] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inf. Softw. Technol.*, vol. 58, pp. 388-402, 2015.
[23] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "Software measurement data reduction using ensemble techniques," *Neurocomputing*, vol. 92, pp. 124-132, 2012.
[24] H. Wang, "A comparative study of ensemble feature selection techniques for software defect prediction," *Mach. Learn. Appl. (ICMLA), Ninth Int. Conf. on. IEEE*, pp. 135-140, 2010.
[25] T. Choeikiwong, and P. Vateekul, "Software defect prediction in imbalanced data sets using unbiased support vector machine," *Inf. Sci. Appl. Springer Berlin Heidelb*, vol. 339, pp. 923-931, 2015.
[26] R. S. Wahono, N. Suryana, and S. Ahmad, "Metaheuristic optimization based feature selection for software defect prediction," *J. Softw.*, vol. 9, no. 5, pp. 1324-1333, 2014.
[27] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, Springer International Publishing, Switzerland, Intelligent Systems Reference Library, vol. 72, pp. 163-193, 2015.
[28] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: a systematic literature review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397-1418, Aug 2013.
[29] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *J. Syst. Softw.*, vol. 83, no. 1, pp. 2-17, 2010.
[30] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," *18th Int. Conf. Eval. Assess. Softw. Eng. ACM*, 2014.
[۳۱] علیقارداشی، *پیش‌بینی اشکال در ماژول‌های نرم‌افزاری با تلفیق روش‌های انتخاب ویژگی در مدل‌های توسعه‌یافته ماشین بردار پشتیبان*، پایان‌نامه کارشناسی ارشد، دانشکده مهندسی برق و کامپیوتر، دانشگاه یزد، یزد، ایران، ۱۳۹۴.
[32] B. Shuai, H. Li, M. Li, Q. Zhang, and C. Tang, "Software defect prediction using dynamic support vector machine," *Comput. Intell. Secur. (CIS), 9th Int. Conf. IEEE*, pp. 260-263, 2013.
[33] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1806-1817, 2012.

زیرنویس‌ها

[1] Fault-prone
[2] Not fault-prone
[3] Forward selection
[4] Ensemble learning
[5] Greedy forward selection
[6] Meta-heuristic optimization
[7] Genetic alghorithm
[8] Particle swarm optimization
[9] Bagging
[10] Symmetric uncertainty
[11] Information gain
[12] Area Under ROC (Receiver Operating Characteristic) Curve
[13] True Positive
[14] False Positive
[15] True Negative
[16] False Negative
[17] True Positive rate
[18] False Positive rate
[19] k-fold cross validation
[20] Smooth Twin Parametric-Margin SVM