Journal of Electronical & Cyber Defence
Vol. 9, No. 3, 2021, Serial No. 35

A model for Feature Selection in Software Fault Prediction Based on the Memetic Algorithm and Fuzzy Logic

A. Karimi[1]*, M. Eshraghi[2], Nia, E. Bastami[3]

*Imam Hossein Comprehensive University, Tehran, Iran

(Received: 01/10/2020, Accepted: 11/01/2021)

ABSTRACT

Today, due to high costs, it is not possible to perform a comprehensive and complete test on all parts of the software. But if the fault-prone parts are identified before the test, the main focus of the test can be placed on these parts, which leads to cost savings. Identifying fault-prone components is the main purpose of software fault prediction. A predictive model receives software modules along with their features as input and predicts which ones are prone to fault. Machine learning techniques are commonly used to construct these models, the performance of which is highly dependent on the training dataset. Training datasets usually have many software features, some of which are irrelevant or redundant, and the removal of these features is done using feature selection methods. In this research, a new method for wrapper-based feature selection is proposed that uses memetic algorithm, random forest technique and a new criterion based on fuzzy inference system. The results show that the proposed fuzzy evaluation criterion has a better performance than the existing criteria and improves the performance of feature selection. The final purpose of this research is to achieve a robust model for predicting high performance software faults and the comparison results show that the proposed model has higher performance than other models.

Keywords: Software Fault Prediction, Feature Selection, Fuzzy Logic, Memetic Algorithm

* Corresponding Author Email: m_Mosavi@iust.ac.ir

نشریه علمی "پدافند الکترونیکی و سایبری"
سال نهم، شماره ۳، پاییز ۱۴۰۰، ص ۱۶۳- ۱۴۳

علمی– پژوهشی

ارائه مدلی برای انتخاب ویژگی در پیش‌بینی خطاهای نرم‌افزار مبتنی بر الگوریتم ممتیک و منطق فازی

علی کریمی*[1]، محمد اشراقی‌نیا[2]، اسماعیل بسطامی[3]

۱- استادیار، عضو هیئت علمی، ۲- دانشجوی کارشناسی ارشد ۳- پژوهشگر و دانشجوی دکتری، دانشگاه جامع امام حسین(ع)

(دریافت: ۱۳۹۹/۱۱/۱۴، پذیرش: ۱۴۰۰/۰۳/۰۸)

چکیده

امروزه به‌دلیل هزینه‌های بالا، انجام آزمون جامع و کامل بر روی تمامی بخش‌های نرم‌افزاری امکان‌پذیر نیست. اما اگر بخش‌های مستعدخطا قبل از انجام آزمون شناسایی شوند، می‌توان تمرکز اصلی آزمون را بر روی این بخش‌ها قرار داد که منجر به صرفه‌جویی در هزینه‌ها می‌شود. شناسایی بخش‌های مستعدخطا، هدف اصلی پیش‌بینی خطا در نرم‌افزار است. یک مدل پیش‌بینی‌کننده، بخش‌های نرم‌افزاری به همراه ویژگی‌های آن‌ها را به عنوان ورودی دریافت کرده و پیش‌بینی می‌کند که کدام یک از آن‌ها مستعدخطا هستند. معمولا برای ساخت این مدل‌ها از فنون یادگیری ماشین استفاده می‌شود که عملکرد این فنون، بسیار وابسته به مجموعه داده آموزشی است. مجمعه داده آموزشی معمولا دارای ویژگی‌های نرم‌افزاری زیادی است که برخی از آن‌ها نامرتبط و یا افزونه بوده و حذف این ویژگی‌ها با استفاده از روش‌های انتخاب ویژگی انجام می‌گردد. در این تحقیق، روش جدیدی برای انتخاب ویژگی مبتنی بر پوشش ارائه شده که از الگوریتم ممتیک، تکنیک جنگل تصادفی و معیار جدید مبتنی بر سیستم استنتاج فازی استفاده می‌کند. نتایج بررسی نشان می‌دهد که معیار ارزیابی فازی ارائه شده، عملکرد بهتری را نسبت به معیارهای موجود داشته و باعث بهبود کارایی انتخاب ویژگی می‌شود. هدف نهایی این تحقیق، رسیدن به یک مدل قدرتمند پیش‌بینی‌کننده خطاهای نرم‌افزاری با کارایی بالا بوده و نتایج مقایسه نشان می‌دهد که مدل ارائه شده، دارای عملکرد و کارایی بالاتری نسبت به دیگر مدل‌ها است.

کلید واژه‌ها: پیش‌بینی خطای نرم‌افزار، انتخاب ویژگی، منطق فازی، الگوریتم ممتیک

۱- مقدمه

امروزه با توجه به رشد نرم‌افزارها از نظر اندازه و پیچیدگی، حفظ کیفیت بالای نرم‌افزار یکی از مشکلات موجود در این حوزه است. کیفیت نرم‌افزار به دلیل وجود خطاهای نرم‌افزاری در زبان برنامه‌نویسی، به طور قابل ملاحظه‌ای کاهش می‌یابد [۱]. تضمین کیفیت نرم‌افزار جنبه‌های مختلفی دارد که یکی از آن‌ها پیش‌بینی خطاهای نرم‌افزاری[1] است. پیش‌بینی خطاهای نرم‌افزاری به معنای شناسایی بخش‌های مستعدخطا در گام‌های اولیه توسعه نرم‌افزار است که باعث کاهش زمان آزمون، تلاش و هزینه می‌شود [۲].

فنون مختلفی برای پیش‌بینی خطا در نرم‌افزار وجود دارد که امروزه فنون یادگیری ماشین جزء پرکاربردترین‌ها در این زمینه هستند [۳]. تحقیقات مختلف نشان داده که کارایی مدل پیش‌بینی‌کننده، وابسته به نوع فن انتخاب شده است [۲].

از طرفی اگر بهترین مدل پیش‌بینی‌کننده نیز وجود داشته باشد اما داده آموزشی نامناسبی به آن وارد شود، نمی‌توان انتظار خروجی مناسبی را از آن داشت. بنابراین انجام پیش‌پردازش بر روی مجموعه داده ورودی، عملی اجتناب‌ناپذیر به نظر می‌رسد.

یکی از مهم‌ترین مسائل در بحث پیش‌پردازش داده، مسئله انتخاب ویژگی است. داده‌ی ورودی معمولا دارای ویژگی‌های زیادی است و انتخاب ویژگی به معنای فرآیند انتخاب زیرمجموعه‌ای از ویژگی‌های مرتبط و مناسب برای ساخت مدل پیش‌بینی‌کننده است. اهمیت این کار از این جهت است که ممکن است نمونه‌ها دارای ویژگی‌های افزونه و یا نامرتبط باشند که به ترتیب، اطلاعات جدید و اطلاعات مفیدی را ارائه نمی‌دهند [۴]. بنابراین حذف آن‌ها می‌تواند منجر به بهبود عملکرد مدل نهایی شود. دو دسته‌بندی کلی برای روش‌های انتخاب ویژگی

* رایانامه نویسنده مسئول: a.karimi@ihu.ac.ir

[1] Software Faults Prediction

ناشر: دانشگاه جامع امام حسین (ع)

وجود دارد که عبارتند از روش‌های مبتنی بر فیلتر[1] و مبتنی بر پوشش[2] [۵]. در این تحقیق، یک روش جدید انتخاب ویژگی مبتنی بر پوشش، برای پیش‌بینی خطا در نرم‌افزار ارائه می‌شود.

۲- ادبیات پژوهش

در این بخش، به بیان برخی از مفاهیم استفاده شده در این تحقیق پرداخته می‌شود.

۲-۱- پیش‌بینی خطا در نرم‌افزار

در بسیاری از مواقع، شناسایی تعداد خطاها در یک بخش نرم‌افزاری اهمیت چندانی ندارد. در واقع آن چیزی که دارای اهمیت است، این است که آیا آن بخش، مستعدخطا هست یا نه [۶]. یک مدل پیش‌بینی‌کننده، یک طبقه‌بند[3] است که بخش‌های نرم‌افزاری را به دو دسته مستعدخطا و غیرمستعدخطا طبقه‌بندی می‌کند. یک مدل، از داده‌های نسخه‌های قبلی پروژه و یا پروژه‌های مشابه استفاده کرده و مستعدخطا بودن و یا نبودن بخش‌های پروژه فعلی را پیش‌بینی می‌کند. این عمل، افزایش بهره‌وری در آزمون و فعالیت‌های تضمین کیفیت نرم‌افزار را به دنبال دارد چرا که باعث می‌شود که تیم آزمون، تمرکز خود را بر روی مولفه‌های مستعدخطا قرار دهند [۷]. این کار آنجایی اهمیت خود را نشان می‌دهد که پروژه دارای محدودیت در بودجه بوده و یا سیستم نرم‌افزاری برای انجام آزمون جامع، بزرگ باشد [۸]. در واقع یکی از روش‌های بهبود کارایی آزمون نرم‌افزار، استفاده از پیش‌بینی خطا در نرم‌افزار است که معمولا بهتر از روش‌های سنتی عمل می‌کند. مثلا احتمال شناسایی بخش خطادار توسط مدل پیش‌بینی‌کننده خطا تقریبا ۷۱ درصد بوده، در حالی که در بازبینی‌های نرم‌افزاری حدود ۶۰ درصد است [۴].

یک مدل پیش‌بینی‌کننده خطا به یک مجموعه داده آموزشی[4] برای یادگیری نیاز دارد. این مجموعه داده شامل سطرها (نمونه‌ها) و ستون‌ها (ویژگی‌ها) است که خطادار بودن و یا نبودن هر نمونه در آن مشخص شده و دارای برچسب است. بنابراین مدل‌های پیش‌بینی‌کننده خطاهای نرم‌افزاری، از نوع مدل‌های یادگیری تحت نظارت[5] هستند. در جدول (۱)، یک مجموعه داده کوچک به عنوان نمونه آورده شده است.

جدول (۱): یک نمونه از مجموعه داده

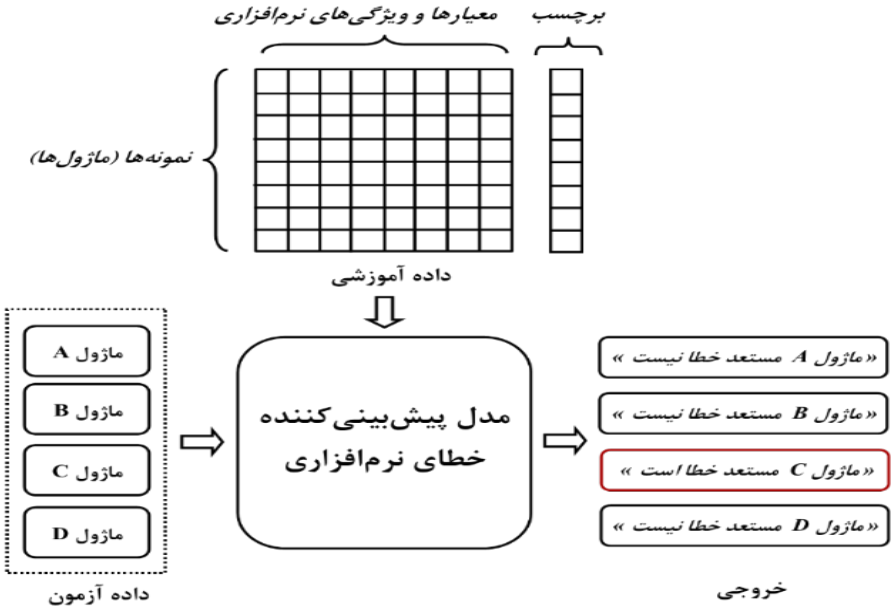
برچسب	...	عمق درخت وراثت	تعداد عملوندها	تعداد عملگرها	تعداد خطوط کد	
خطادار	...	۲	۲۶	۱۵	۴۵	بخش ۱
بدون خطا	...	۳	۱۹	۹	۷۶	بخش ۲
بدون خطا	...	۱	۱۷	۱۲	۵۲	بخش ۳
خطادار	...	۱	۳۰	۱۹	۱۳۱	بخش ۴
بدون خطا	...	۲	۴۵	۲۵	۹۸	بخش ۵
...	...	...	...	...	...	...

یک مدل پیش‌بینی‌کننده بر اساس معیارها و ویژگی‌های نرم‌افزاری، ایجاد شده و پس از یادگیری می‌تواند مستعدخطا بودن و یا نبودن هر نمونه ورودی را تشخیص دهد. هرچه مجموعه داده یادگیری و مدل پیش‌بینی‌کننده مناسب‌تر باشند، قدرت تشخیص و کارایی مدل بالاتر می‌رود. نحوه عملکرد کلی مدل پیش‌بینی‌کننده خطا در شکل (۱) نشان داده شده است.

۲-۲- معیارهای نرم‌افزاری

برای هر بخش نرم‌افزاری، معیارها و ویژگی‌های مختلفی تعریف شده است که هر کدام، یک جنبه‌ی خاص از آن بخش را توصیف می‌کنند. هر کدام از این معیارها، برای ارزیابی کیفیت نرم‌افزار نیز مورد استفاده قرار می‌گیرد. سیستم‌های پیش‌بینی خطا، با استفاده از معیارهای نرم‌افزاری، مستعدخطا بودن و یا نبودن نمونه‌ها را پیش‌بینی می‌کنند. بنابراین معیارهای نرم‌افزاری به عنوان متغیرهای مستقل (متغیرهای پیشگو) و مستعدخطا بودن و نبودن نمونه‌ها به عنوان متغیرهای وابسته شناخته می‌شوند [۴].

[1] Filter based
[2] Wrapper based
[3] Classifier
[4] Training dataset
[5] Supervised learning



شکل (۱): نحوه عملکرد مدل پیش‌بینی‌کننده‌خطا

یکی از مجموعه داده‌های معروف در حوزه پیش‌بینی خطای نرم‌افزار، مجموعه داده NASA بوده که در بخش ارزیابی روش پیشنهادی نیز مورد استفاده قرار گرفته است. این مجموعه داده از معیارهای کد و پیچیدگی استفاده کرده که توصیف مختصر برخی از این معیارها در جدول (۲) آورده شده است.

جدول (۲): توصیف برخی از معیارهای کد و پیچیدگی [۴]

معیار	توضیحات
v(g)	پیچیدگی سیکلوماتیک: برابر است با E-N+2 جایی که E برابر تعداد لبه‌های گراف جریان و N برابر تعداد گره‌های آن است
ev(g)	پیچیدگی اساسی: علامت‌گذاری کدهای بدون ساختار درون یک بخش است که به منظور محاسبه پیش‌بینی تلاش برای نگهداری بخش استفاده می‌شود
iv(g)	پیچیدگی طراحی: تعداد فراخوانی‌هایی که مستقیما توسط یک بخش انجام می‌شود یا تعداد بخش‌هایی که به طور مستقیم یک بخش را فراخوانی می‌کنند
length	برابر است با N1+N2 جایی که N1 برابر با تعداد عملوندهای بخش و N2 برابر با تعداد عملگرهای آن است
volume	برابر است با $length \times \log_2(\varphi1+\varphi2)$ جایی که $\varphi1$ برابر با تعداد عملگرهای متمایز و $\varphi2$ برابر با تعداد عملوندهای متمایز است
difficulty	برابر است با $(\frac{\varphi1}{2}) \times (\frac{N2}{\varphi2})$ که این معیار میزان درک و فهم بخش را مشخص می‌کند
content	برابر $level \times volume$ است که در آن $level$ یک برنامه، مقداری بین ۰ و ۱ را داراست که سطح ۱ نشان‌دهنده آن است که یک بخش در بالاترین سطح ممکن تشکیل شده است یعنی با حداقل اندازه

یکی دیگر از مجموعه داده‌های معروف در حوزه پیش‌بینی خطای نرم‌افزار، مجموعه داده PROMISE می‌باشد. این مجموعه داده نیز در بخش ارزیابی روش پیشنهادی مورد استفاده قرار گرفته و در مجموع دارای ۲۰ معیار نرم‌افزاری شیءگرایی است. توصیف مختصر این معیارها در جدول (۳) آورده شده است.

جدول (۳): توصیف برخی از معیارهای شیءگرایی [۳]

معیار	توضیحات
wmc	تعداد متدهای تعریف شده در یک کلاس
dit	حداکثر ارتفاع سلسله مراتب کلاس از ریشه وراثت
noc	تعداد فرزندان فوری یک کلاس
cbo	تعداد کلاس‌هایی که متصل به کلاس مورد نظر هستند
rfc	تعداد متدهای مختلفی که می‌توانند هنگام دریافت پیام از یک شیء، از آن کلاس اجرا شوند
lcom	تعداد مجموعه‌ای از متدها در یک کلاس که از طریق به اشتراک‌گذاری برخی از بخش‌های کلاس با هم مرتبط نیستند
ca	تعداد کلاس‌های وابسته به یک کلاس مشخص
ce	تعداد کلاس‌هایی که یک کلاس به آن‌ها وابستگی دارد
npm	تعداد متدهای عمومی (public) که در یک کلاس تعریف شده است

۲-۳- انتخاب ویژگی مبتنی بر پوشش

روش‌های مبتنی بر پوشش بر خلاف روش‌های فیلتری، به طور مستقیم از الگوریتم‌های یادگیری به منظور ارزیابی مجموعه ویژگی‌ها استفاده می‌کنند. در این روش‌ها، فرآیند انتخاب ویژگی یک ترکیب تکراری از سه بخش جستجو، یادگیری و ارزیابی است [۹]. ابتدا با استفاده از روش‌های جستجو (مانند الگوریتم ژنتیک)، چندین زیرمجموعه از ویژگی‌ها به عنوان پاسخ‌های کاندید (مجموعه ویژگی‌های بهینه) انتخاب می‌شوند. به عنوان مثال مجموعه A={1,3,7,8} می‌تواند به عنوان یک پاسخ کاندید ارائه شود و بدین معناست که فقط ویژگی‌های شماره ۱، ۳، ۷ و

۸ ویژگی‌های بهینه هستند و باقی ویژگی‌ها باید از مجموعه داده حذف شوند. سپس برای ارزیابی میزان خوب بودن مجموعه A یا همان مقدار برازش[1] الگوریتم جستجو، مجموعه داده‌های آموزشی و آزمون، بر اساس ویژگی‌های موجود در مجموعه A فیلتر می‌شوند. در واقع تنها ویژگی‌های شماره ۱، ۳، ۷ و ۸ در مجموعه داده‌های آموزشی و آزمون باقی می‌مانند و دیگر ویژگی‌ها از آن‌ها حذف می‌شوند. سپس یک طبقه‌بند (مثلا درخت تصمیم) با استفاده از داده آموزشی فیلترشده، ساخته می‌شود. بنابراین تنها ویژگی‌های موجود در آن کاندید در ساخت طبقه‌بند تاثیرگذارند و دیگر ویژگی‌ها در ساخت آن تاثیری ندارند. در مرحله بعد، مجموعه داده آزمون فیلترشده به طبقه‌بند داده می‌شود و کارایی آن (مثلا صحت[2]) اندازه‌گیری شده و این مقدار صحت، بیانگر همان مقدار برازش کاندید A است. بنابراین کاندیدی که طبقه‌بند متناظر با آن، مقدار صحت بالاتری را داشته باشد، دارای ویژگی‌های بهینه‌تری است و در مجموع کاندید بهتری نسبت به بقیه است. این فرآیند تولید زیرمجموعه‌های کاندید توسط الگوریتم جستجو و محاسبه مقدار برازش آن‌ها، آنقدر تکرار می‌شود تا شرط پایان تکرار الگوریتم جستجو محقق شود و در نهایت، بهترین مجموعه ویژگی‌ها از لحاظ بالاترین صحت طبقه‌بندی به دست می‌آید که همان مجموعه ویژگی‌های بهینه و موردنظر هستند. در واقع این روش‌ها از بازخورد الگوریتم یادگیری و یا همان طبقه‌بند استفاده کرده تا مشخص کنند که کدام ویژگی‌ها باید برای ساخت مدل پیش‌بینی‌کننده انتخاب شوند [۱۰]. به دلیل استفاده از طبقه‌بند، این روش‌ها بهتر از روش‌های فیلتری عمل کرده و دقت پیش‌بینی بالاتری را به دنبال دارند [۳, ۹, ۱۱, ۱۲].

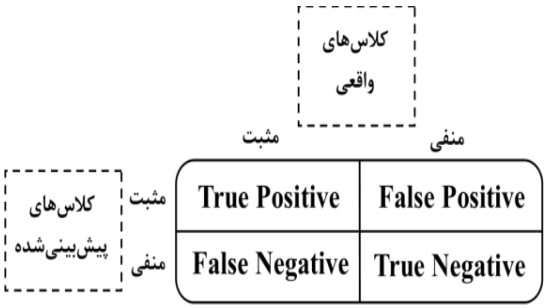
۲-۴- الگوریتم ممتیک

الگوریتم ممتیک، یک بهبود از الگوریتم ژنتیک است [۱۴]. این الگوریتم، حاصل ترکیب الگوریتم ژنتیک و عملگرهایش با جستجوی محلی است و به همین دلیل، به الگوریتم ژنتیک ترکیبی[3] نیز معروف است [۱۵]. در واقع در هر مرحله که جمعیت جدیدی تولید می‌شود، افراد آن جمعیت به جستجوی همسایگان اطراف خود می‌پردازند. اگر برای هر کروموزوم، همسایه‌ای بهتر از خودش پیدا شد، آن همسایه جایگزین آن خواهد شد. بنابراین پس از تولید هر نسل، خود آن نسل نیز تلاش برای بهتر شدن می‌کند. در واقع وجود جستجوی محلی همراه با عملگر جهش، بهره‌برداری مورد نیاز را تضمین می‌کند و عملگر تلاقی نیز منجر به اکتشاف شده و به ایجاد تعادل کمک می‌کند. در نتیجه الگوریتم ممتیک به یک الگوریتم فرااکتشافی بسیار مفید تبدیل شده است [۱۴].

[1] Fitness
[2] Accuracy
[3] Hybrid genetic algorithm

۲-۵- معیارهای ارزیابی کارایی طبقه‌بند

یک مدل طبقه‌بند، پس از یادگیری و آموزش، نیاز به ارزیابی دارد. این کار با اجرای مجموعه داده‌ی آزمون بر روی طبقه‌بند انجام می‌پذیرد. برای ارزیابی یک طبقه‌بند و مقایسه عملکرد آن با دیگر طبقه‌بندها، به یک معیار کمّی و مشخص نیاز است. برای معرفی معیارهای ارزیابی، ابتدا باید ماتریس درهم‌ریختگی[4] معرفی شود. این ماتریس در شکل (۲) نشان داده شده است.



شکل (۲): ماتریس درهم‌ریختگی

۲-۵-۱- صحت

این معیار از طریق رابطه (۱) محاسبه می‌شود [۱۶]:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (۱)$$

۲-۵-۲- نرخ خطا[5]

مکمل معیار صحت، معیار نرخ خطا است که در رابطه (۲) آورده شده است:

$$EER = 1 - Acc = \frac{FP + FN}{TP + TN + FP + FN} \quad (۲)$$

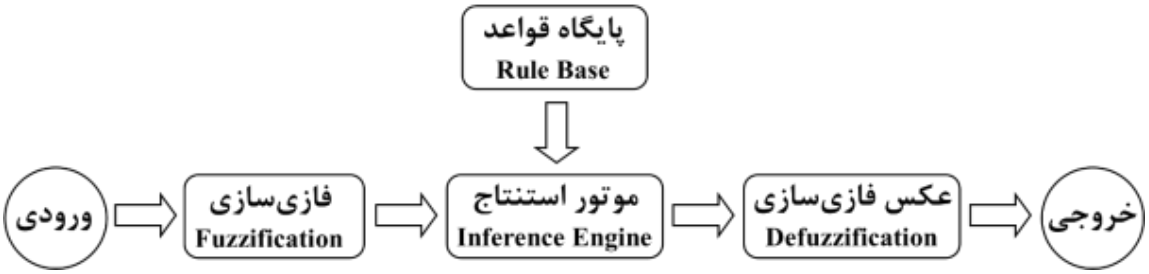
هر دو معیار صحت و نرخ خطا به مجموعه‌داده‌های نامتعادل حساس هستند [۱۶].

۲-۵-۳- فراخوانی[6]

فراخوانی از رابطه (۳) به‌دست می‌آید [۱۷] و نام دیگر آن نرخ مثبت صحیح[7] است [۱۶].

$$Recall = TPR = \frac{TP}{TP + FN} \quad (۳)$$

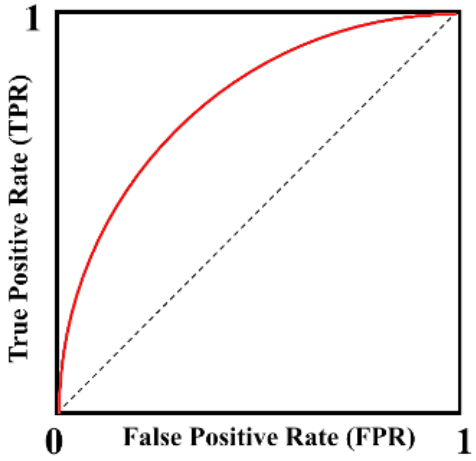
[4] Confusion matrix
[5] Error rate
[6] Recall
[7] True Positive Rate



شکل (۴): ساختار کلی سیستم استنتاج فازی

۲-۵-۴- مساحت زیر منحنی ROC [1]

ROC یک منحنی بر روی یک صفحه دوبعدی است که محور y آن را نرخ مثبت صحیح و محور x آن را نرخ مثبت کاذب [2] تشکیل می‌دهد. شکل (۳) یک نمونه از منحنی ROC را نمایش می‌دهد که مساحت زیر این منحنی، بیانگر همان مقدار AUC است.



شکل (۳): منحنی ROC

۲-۶- سیستم استنتاج فازی

در منطق فازی، برای توصیف مقادیر از متغیرهای زبانی استفاده می‌شود، مثل متغیر زبانی «خوب» و یا «عالی». مقدار کیفی یک متغیر زبانی را می‌توان با مجموعه‌های مختلف فازی و توابع عضویت مخصوص، بیان کرد [۱۷]. سیستم استنتاج فازی (FIS) [3] با استفاده از منطق فازی، فضای ورودی را به فضای خروجی نگاشت می‌کند.

منطق فازی از چهار بخش کلی ۱- پایگاه قواعد، ۲- فازی‌سازی، ۳- موتور استنتاج و ۴- عکس فازی‌سازی تشکیل می‌شود که ساختار آن در شکل (۴) آورده شده است.

[1] AUC
[2] False Positive Rate
[3] Fuzzy Inference System

۳- کارهای مرتبط

روش‌های انتخاب ویژگی مبتنی بر پوشش به دلیل استفاده از الگوریتم یادگیری، دقت طبقه‌بندی بالاتری را نسبت به روش‌های مبتنی بر فیلتر به همراه دارند [۳, ۲۰]. ترابیه [4] و همکارانش از یک روش مبتنی بر پوشش برای پیش‌بینی خطاهای نرم‌افزاری استفاده کرده‌اند [۳]. فاریس [5] و همکارانش در تحقیق خود دو روش جدید مبتنی بر پوشش را ارائه کرده که نتایج کار، نشان از ایجاد بهبود در مسئله انتخاب ویژگی دارد [۲۱]. تومار [6] و همکاران نیز یک روش مبتنی بر پوشش برای انتخاب ویژگی در پیش‌بینی خطای نرم‌افزار ارائه کرده‌اند که روش آن‌ها باعث بهبود کارایی طبقه‌بند شده است [۲۲]. ترابیه و همکاران [۳] در روش انتخاب ویژگی مبتنی بر پوشش خود از الگوریتم‌های ژنتیک، ازدحام ذرات و کلونی مورچگان برای جستجو در فضای ویژگی‌ها استفاده کرده و همچنین برای ارزیابی ویژگی‌ها، طبقه‌بند شبکه عصبی به کار گرفته شده است. آن‌ها روش خود را بر روی مجموعه داده PROMISE ارزیابی کرده و با کارهای ارتورک [7] [۱۹] و شاتناوی [8] [۲۳] مقایسه کرده‌اند که در مجموع مدل آن‌ها عملکرد بهتری را از خود نشان داده است.

۳-۱- روش‌های جستجوی ویژگی‌ها

جستجو در فضای ویژگی‌ها به منظور یافتن ویژگی‌های بهینه، یک مسئله ان‌پی- سخت است [۱] و معمولا برای این منظور از روش‌های فرااکتشافی استفاده می‌شود. در جدول (۴) خلاصه‌ای از برخی الگوریتم‌های فرااکتشافی استفاده شده در موضوع انتخاب ویژگی آورده شده است.

[4] Turabieh
[5] Faris
[6] Tumar
[7] Erturk
[8] Shatnawi

جدول (۴): الگوریتم‌های فرااکتشافی استفاده شده در انتخاب ویژگی

ژنتیک	راجکومار[1] و ویجی[2] [۱] الاکیا[3] و سلواکومار[4] [۱۳]
ممتیک	ژو[5] و همکاران [۱۲] کبیر[6] و همکاران [۲۴] لیو[7] و همکاران [۱۱] ابوظاهر[8] و همکاران [۹] لی[9] و همکاران [۲۵]
سایر	چاکرابورتی[10] [۲۶]: ازدحام ذرات فاریس و همکاران: ازدحام سالپ[11] [۲۱] باردامووا[12] و همکاران [۲۷]: جستجوی گرانشی تومار و همکاران [۲۲]: شمع و پروانه[13]
چندین الگوریتم	ترابیه و همکاران [۳]: ژنتیک، ازدحام ذرات، کلونی مورچگان یوستا[14] [۱۵]: ژنتیک، ممتیک، جستجوی ممنوعه و GRASP چاکرابورتی و کاوامورا[15] [۲۸]: ژنتیک، ازدحام ذرات، جستجوی گرانشی، جستجوی فاخته، جستجوی کرم شب‌تاب، جستجوی خفاش، جستجوی سنجاقک

۳-۲- طبقه‌بندهای استفاده شده

بابامیر و کریمیان [۴] در پژوهش خود به مقایسه ۱۷ طبقه‌بند مختلف در پیش‌بینی خطای نرم‌افزار پرداخته‌اند. آن‌ها آزمایش خود را بر اساس ۴ روش مختلف بر روی مجموعه داده NASA انجام داده و عملکرد هر طبقه‌بند را بر اساس معیار AUC مورد ارزیابی قرار داده‌اند. در نهایت سه طبقه‌بند Decorate، Bagging و Random Forest توانستند که بهترین میانگین AUC را کسب کنند. کالسوم[16] و همکاران [۲۹] در کار خود به مقایسه عملکرد ۹ طبقه‌بند مهم و پراستفاده در حوزه پیش‌بینی خطای نرم‌افزار پرداخته‌اند و در نهایت سه طبقه‌بند Naïve Bayes، MLP و Random Forest توانستند عملکرد خوب و قابل قبولی را در تمامی مجموعه داده‌ها از خود نشان بدهند.

[1] Rajkumar
[2] Viji
[3] Elakkiya
[4] Selvakumar
[5] Zhu
[6] Kabir
[7] Liu
[8] Abu Zaher
[9] Lee
[10] Chakraborty
[11] Salp swarm
[12] Bardamova
[13] Moth flame
[14] Yusta
[15] Kawamura
[16] Kalsoom

همانطور که بیان شد، در انتخاب ویژگی مبتنی بر پوشش به منظور ارزیابی ویژگی‌ها نیز از طبقه‌بند استفاده می‌شود. خلاصه‌ای از طبقه‌بندهای استفاده شده در انتخاب ویژگی مبتنی بر پوشش در جدول (۵) آورده شده است.

جدول (۵): طبقه‌بندهای استفاده شده در انتخاب ویژگی مبتنی بر پوشش

SVM	راجکومار و ویجی [۱] چاکرابورتی و کاوامورا [۲۸] گویون[17] و همکاران [۳۰]
Neural Network	ترابیه و همکاران [۳]: L-RNN کبیر و همکاران [۲۴]
KNN	ژو و همکاران [۱۲]: 1-NN فاریس و همکاران [۲۱] ژو[18] و همکاران [۳۱] چوانگ[19] و همکاران [۳۲] تومار و همکاران [۲۲]
سایر	پاتچایامال[20] و تیرومالایسلوی[21] [۳۳]: طبقه‌بند خطی لیو و همکاران [۱۱]: LR الاکیا و سلواکومار [۱۳]: RF برمجو[22] و همکاران [۳۴]: NB قوش[23] و همکاران [۱۴]: MLP، SVM، KNN

۳-۳- معیارهای ارزیابی طبقه‌بند

همانطور که در بخش (۲-۳) بیان شد، برای ارزیابی ویژگی‌ها در انتخاب ویژگی مبتنی بر پوشش، ابتدا یک طبقه‌بند با استفاده از آن ویژگی‌ها ساخته شده و سپس با استفاده از یک معیار ارزیابی مانند صحت، عمل ارزیابی طبقه‌بند انجام می‌شود. در ادامه به بیان معیارهای ارزیابی طبقه‌بند مورد استفاده در این بخش، یعنی انتخاب ویژگی مبتنی بر پوشش پرداخته می‌شود.

راجکومار و ویجی از یک فرمول که ترکیبی از خطای طبقه‌بند و تعداد ویژگی‌های انتخاب شده است، برای ارزیابی ویژگی‌ها استفاده کرده‌اند [۱]. ترابیه و همکاران فرمولی متشکل از نرخ خطا و تعداد ویژگی‌های انتخاب شده را به عنوان تابع برازش ارائه کرده‌اند [۳]. در تحقیق فاریس و همکاران از نرخ

[17] Guyon
[18] Xue
[19] Chuang
[20] Patchaiammal
[21] Thirumalaiselvi
[22] Bermejo
[23] Ghosh

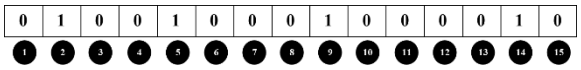
خطای طبقه‌بندی و تعداد ویژگی‌های انتخاب شده برای ارزیابی ویژگی‌های انتخاب شده استفاده شده است [۲۱]. در تحقیق قوش و همکاران، معیار صحت به عنوان هدف اول و سپس تعداد ویژگی‌های انتخاب شده به عنوان هدف دوم در تابع برازش الگوریتم ممتیک در نظر گرفته شده است [۱۴]. الاکیا و سلواکومار از نرخ خطای جنگل تصادفی برای ارزیابی ویژگی‌ها استفاده کرده‌اند [۱۳]. پاسکوال[1] و همکارانش در تحقیق خود اشاره کرده‌اند که در روش‌های مبتنی بر پوشش، ویژگی‌ها بر اساس معیارهای کارایی طبقه‌بند از قبیل دقت و فراخوانی انتخاب می‌شوند [۳۵]. تومار و همکاران نیز یک فرمول خاص مبتنی بر نرخ خطای طبقه‌بند و تعداد ویژگی‌های انتخاب شده برای این منظور ارائه کرده‌اند [۲۲]. ژو و همکاران از معیار صحت طبقه‌بندی برای ارزیابی ویژگی‌ها استفاده کرده‌اند [۱۲]. کبیر و همکاران از جمع صحت طبقه‌بند شبکه عصبی و یک مقدار مجزا به عنوان تابع ارزیاب ویژگی‌ها استفاده کرده‌اند [۲۴]. لیو و همکاران از صحت طبقه‌بند LR برای تابع برازش انتخاب ویژگی استفاده کرده‌اند [۱۱]. چاکرابورتی و همچنین یوستا در تحقیق خود به طور جداگانه به این موضوع اشاره کرده‌اند که در روش‌های مبتنی بر پوشش، زیرمجموعه بهینه از ویژگی‌ها بر اساس معیار صحت طبقه‌بند به دست می‌آید [۱۵, ۲۶]. چاکرابورتی و کاوامورا برای ارزیابی ویژگی‌ها، معیاری مبتنی بر جریمه را ارائه کرده‌اند که ترکیبی از معیار صحت طبقه‌بند SVM و تعداد ویژگی‌های انتخاب شده به عنوان جریمه است [۲۸]. خونجی[2] و همکاران و همچنین رانی[3] و همکاران نیز در تحقیق خود به این موضوع اشاره داشته‌اند که در روش‌های مبتنی بر پوشش، ویژگی‌ها بر اساس معیار صحت طبقه‌بندی انتخاب می‌شوند [۲۰, ۳۶]. کومار و همکاران نیز معیار خوب بودن ویژگی‌ها در روش‌های مبتنی بر پوشش را صحت طبقه‌بندی می‌دانند [۳۷]. ژو و همکاران، آن دسته از ویژگی‌هایی را به عنوان ویژگی‌های منتخب انتخاب می‌کنند که نرخ خطای طبقه‌بندی را به حداقل برسانند [۳۱].

۴- روش انتخاب ویژگی پیشنهادی (MRF-FS)[4]

روش انتخاب ویژگی ارائه شده برگرفته از روش انتخاب ویژگی مبتنی بر پوشش است که در بخش (۳-۲) معرفی شده است. همانطور که در شکل (۵) مشخص است، ورودی انتخاب ویژگی، مجموعه داده مورد نظر است و خروجی آن نیز، همان مجموعه داده اما با ویژگی‌های بهینه می‌باشد.

شکل (۵): ورودی و خروجی انتخاب ویژگی

مجموعه ویژگی‌ها به صورت یک رشته دودویی کدگذاری می‌شوند که طول این رشته برابر با تعداد کل ویژگی‌ها است. مثلا اگر مجموعه دارای ۱۵ ویژگی باشد و فقط ویژگی‌های شماره ۲، ۵، ۹ و ۱۴ ویژگی‌های بهینه باشند، این زیرمجموعه بهینه از ویژگی‌ها به صورت شکل (۶) نمایش داده می‌شود.



شکل (۶): مثالی از کدگذاری ویژگی‌ها در انتخاب ویژگی

در واقع باید ویژگی‌های اضافه از مجموعه داده حذف شده و فقط ستون‌های مربوط به ویژگی‌های بهینه در آن باقی بمانند. اما یافتن این مجموعه بهینه از ویژگی‌ها یک مسئله ان‌پی-سخت است و برای به دست آوردن آن معمولا از الگوریتم‌های فرااکتشافی استفاده می‌شود. برای مثال فرض می‌شود که از الگوریتم ژنتیک برای این کار استفاده شده و اندازه جمعیت در این الگوریتم نیز ۵ در نظر گرفته شده است. بنابراین در ابتدا ۵ کروموزوم که جواب‌های کاندید مسئله هستند، به صورت تصادفی و مطابق شکل (۷) مقداردهی اولیه می‌شوند که همان جمعیت اولیه هستند.

سپس مطابق با الگوریتم ژنتیک، عمل انتخاب، تلاقی و جهش انجام می‌شود و نسل جدیدی را به وجود می‌آورد که نسبت به نسل قبلی تکامل یافته‌اند. این فرآیند آنقدر تکرار می‌شود تا شرط پایان الگوریتم ژنتیک که همان تعداد تکرار مشخصی است، محقق شود. در نهایت آن کروموزوم و یا پاسخی که نسبت به بقیه بهتر بوده است، به عنوان پاسخ نهایی در نظر گرفته شده و مجموعه ویژگی‌های بهینه و یا نزدیک به بهینه به دست می‌آید. اما نکته‌ای که در انتخاب ویژگی مبتنی بر پوشش مطرح است، تابع برازش الگوریتم جستجو و یا همان ژنتیک است. در این نوع انتخاب ویژگی، از یک طبقه‌بند برای محاسبه مقدار برازش استفاده می‌شود. به عنوان مثال قرار است که مقدار برازش کروموزوم A در شکل (۷) مورد محاسبه قرار گیرد. همانطور که مشخص است، در کروموزوم A تنها ویژگی‌های شماره ۱، ۵، ۸، ۱۱ و ۱۳ انتخاب شده‌اند. بنابراین تنها این ویژگی‌ها در مجموعه داده آموزشی و آزمون، نگه داشته شده و باقی ویژگی‌ها از آن حذف می‌شود که در نتیجه، مجموعه داده‌ای مشابه با جدول (۶) به دست می‌آید.

[1] Pascoal
[2] Khonji
[3] Rani
[4] Memetic-Random forest-Fuzzy Feature Selection

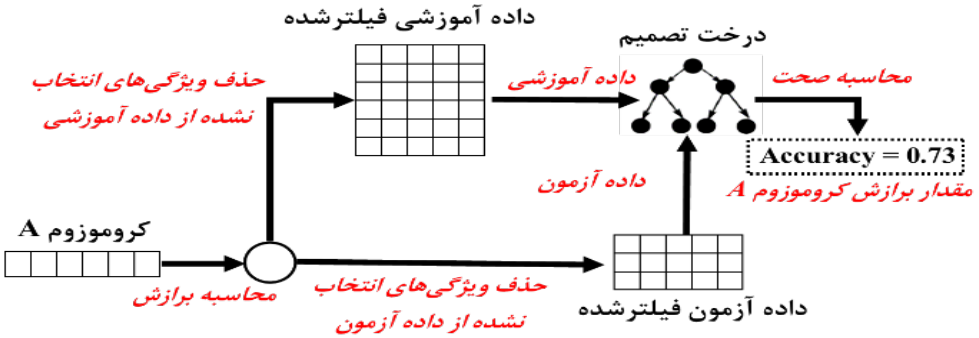
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
کروموزوم A	1	0	0	0	1	0	0	1	0	0	1	0	1	0	0
کروموزوم B	0	1	0	0	1	1	1	0	0	1	0	0	0	1	1
کروموزوم C	0	0	1	1	0	0	1	0	1	0	1	0	1	1	1
کروموزوم D	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0
کروموزوم E	1	0	1	1	0	0	1	0	1	1	0	1	0	1	0

شکل (۷): جمعیت اولیه در الگوریتم ژنتیک

جدول (۶): مجموعه داده با ویژگی‌های فیلترشده

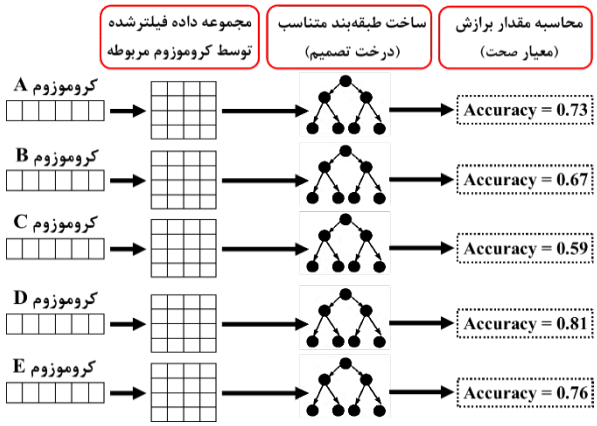
برچسب	ویژگی ۱۳	ویژگی ۱۱	ویژگی ۸	ویژگی ۵	ویژگی ۱	
خطادار	۹/۴	۱۷	۷۸	۱/۴	۵	بخش ۱
بدون خطا	۱۳/۵	۱۲	۹۵	۲/۲	۳	بخش ۲
بدون خطا	۷/۶	۱۵	۱۲۵	۲/۳	۲	بخش ۳
خطادار	۸/۹	۱۳	۸۹	۱/۸	۸	بخش ۴
...	...	...	...	...	...	...

آنگاه یک طبقه‌بند (مثلا درخت تصمیم) با استفاده از مجموعه داده آموزشی جدید که ویژگی‌های آن مطابق با کروموزوم A فیلتر شده‌اند، ایجاد می‌شود. سپس مجموعه داده آزمون فیلترشده نیز به طبقه‌بند وارد شده و یکی از معیارهای ارزیابی (به عنوان مثال صحت) آن محاسبه می‌شود. این مقدار صحت همان مقدار برازش کروموزوم A است. فرآیند کلی محاسبه برازش برای کروموزوم A در شکل (۸) آورده شده است.



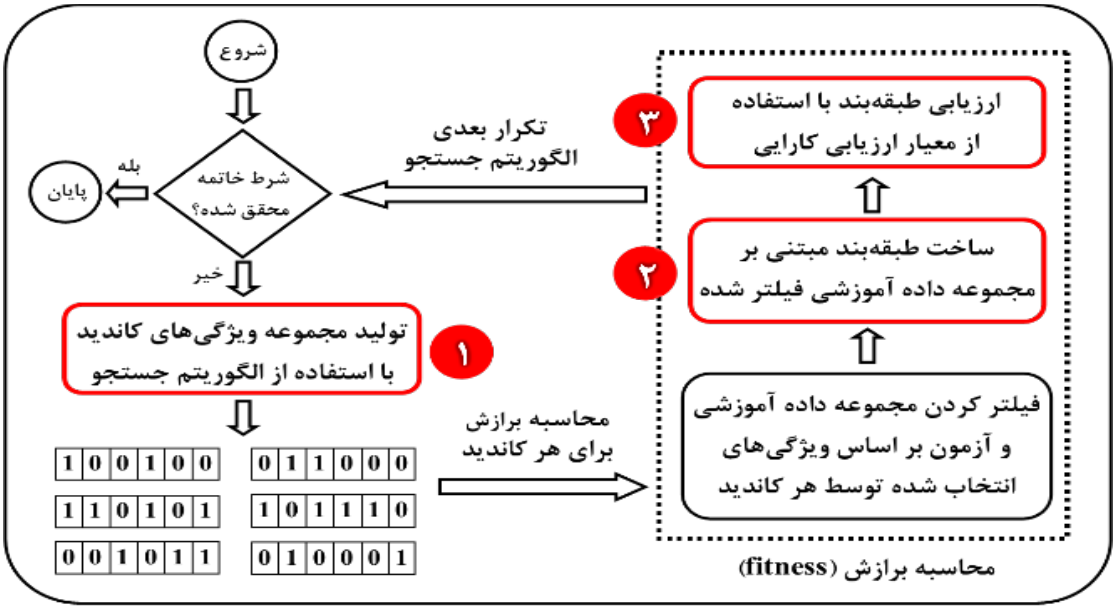
شکل (۸): فرآیند محاسبه مقدار برازش یک کروموزوم

بنابراین مقدار برازش هر ۵ کروموزوم موجود در شکل (۷) که همان جمعیت اولیه الگوریتم ژنتیک هستند، مطابق شکل (۹) محاسبه می‌شود.



شکل (۹): نحوه محاسبه برازش جمعیت اولیه

همانطور که در شکل (۱۰) نشان داده شده است، روش انتخاب ویژگی مبتنی بر پوشش، دارای ۳ بخش اساسی و مهم است. بخش اول آن، انتخاب الگوریتم جستجوی فرااکتشافی مناسب است که وظیفه جستجو در فضای ویژگی‌ها را بر عهده داشته و در مثال ذکرشده، از الگوریتم ژنتیک استفاده شده بود. بخش‌های دوم و سوم، مربوط به نحوه محاسبه برازش در الگوریتم جستجوی انتخابی است. در واقع بخش دوم، انتخاب همان طبقه‌بندی است که در محاسبه برازش هر کروموزوم، بر اساس ویژگی‌های انتخاب شده آن کروموزوم ساخته می‌شود که در مثال ذکرشده، از طبقه‌بند درخت تصمیم استفاده شده بود. بخش سوم نیز انتخاب معیار ارزیابی کارایی طبقه‌بند ساخته شده در بخش قبلی است که در مثال ذکرشده، از معیار صحت استفاده شده بود.



شکل (۱۰): بخش‌های اصلی انتخاب ویژگی مبتنی بر پوشش

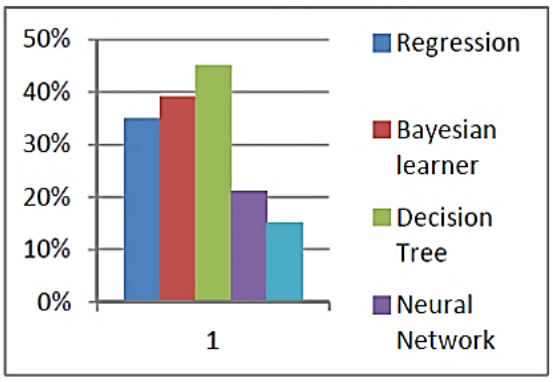
برای بهبود عملکرد انتخاب ویژگی، باید هر یک از این سه بخش به درستی انتخاب شود. در ادامه، نحوه انتخاب هر یک از این سه بخش در روش انتخاب ویژگی ارائه شده، توضیح داده شده است.

۴-۱- بخش ۱، انتخاب الگوریتم جستجوی مناسب

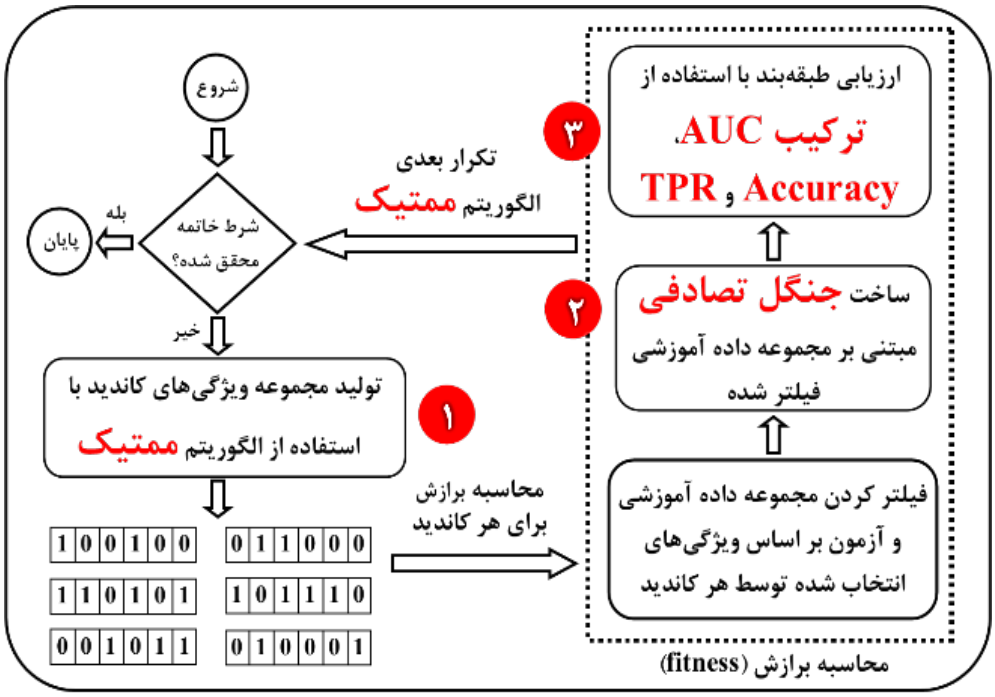
در ادبیات تحقیق معمولا از الگوریتم‌های فرااکتشافی برای جستجو در فضای ویژگی‌ها به منظور یافتن مناسب‌ترین ویژگی‌ها استفاده شده است. طبق گفته یوستا، بیش‌ترین الگوریتم فرااکتشافی استفاده شده در موضوع انتخاب ویژگی، الگوریتم ژنتیک است [۱۵]. یکی از معیارهای یک الگوریتم فرااکتشافی خوب، ایجاد تعادل مناسب بین اکتشاف و بهره‌برداری است. الگوریتم ژنتیک قابلیت اکتشاف بالایی دارد اما در بهره‌برداری دارای نقاط ضعفی است. الگوریتم ممتیک این نقطه ضعف را بهبود داده و با اضافه کردن جستجوی محلی، موجب دستیابی به بهره‌برداری بهتر در فضای جستجو می‌شود. بنابراین الگوریتم ممتیک یک بهبود از الگوریتم ژنتیک است [۱۴] و آزمایشات انجام شده بر روی چندین مجموعه داده، نشان از برتری ممتیک نسبت به ژنتیک دارند [۹]. مطالعات انجام شده بر روی الگوریتم ممتیک تایید کرده است که برای مسائل ان‌پی- سخت مانند مسئله انتخاب ویژگی، این الگوریتم‌ها بهتر از الگوریتم‌های تکاملی سنتی می‌توانند فضای جستجو را کشف کنند [۹]. بنابراین در روش انتخاب ویژگی پیشنهادی، جستجو در فضای ویژگی‌ها با استفاده از الگوریتم ممتیک انجام می‌پذیرد.

۴-۲- بخش ۲، انتخاب طبقه‌بند مناسب

معمولا برای ایجاد طبقه‌بند در انتخاب ویژگی مبتنی بر پوشش از تکنیک‌های یادگیری ماشین استفاده می‌شود. طبق تحقیقات، بیش‌ترین تکنیک مورد استفاده در سالیان اخیر در حوزه پیش‌بینی خطای نرم‌افزار مربوط به روش‌های مبتنی بر درخت تصمیم است [۲] که در شکل (۱۱) نشان داده شده است. طبقه‌بند جنگل تصادفی از مجموعه‌ی درختان تصمیم تشکیل شده و عملکرد قابل قبولی را از خود نشان داده و در برخی پژوهش‌ها، برتری طبقه‌بندی قابل توجهی به آن نسبت داده شده است [۳۸]. همچنین طبق پژوهش‌های [۴, ۸] جنگل تصادفی دارای عملکرد پایدار و قابل اطمینانی بوده است. بنابراین در روش پیشنهادی از جنگل تصادفی برای ایجاد مدل طبقه‌بند استفاده می‌شود. همچنین قبل از ایجاد طبقه‌بند به منظور متعادل‌سازی داده‌ها، تکنیک نمونه‌برداری Resample به‌کار گرفته می‌شود.



شکل (۱۱): روش‌های یادگیری ماشین استفاده شده [۲]



شکل (۱۲): بخش‌های انتخاب ویژگی MRF

۴-۳- بخش ۳، معیار ارزیابی در انتخاب ویژگی

تاکنون معیارهای مختلفی برای ارزیابی کارایی طبقه‌بندها معرفی شده است. در بحث انتخاب ویژگی مبتنی بر پوشش که از طبقه‌بند استفاده می‌شود، این مقدار ارزیابی کارایی به عنوان تابع برازش الگوریتم جستجو تعریف می‌شود و همانطور که در بخش (۳-۳) مشاهده شد، معمولا از معیار صحت و یا مکمل آن یعنی نرخ خطا برای این منظور استفاده شده است. به‌طور کلی هر معیار ارزیابی از یک بعد و جنبه خاص قادر به ارزیابی مدل است و بالاتر بودن امتیاز یک مدل در یک معیار خاص لزوما به معنای بهتر بودن عملکرد آن مدل نیست. نتایج انتخاب ویژگی با توجه به در نظر گرفتن تمامی معیارها متفاوت است، به طوری که هیچ یک از روش‌ها در تمامی معیارها به نتیجه نمی‌رسند. بنابراین برای انتخاب ویژگی باید چندین معیار دخیل باشد و در نظر گرفته شود [۳۹]. در این پژوهش نیز به جای یک معیار از ترکیب چندین معیار برای ارزیابی طبقه‌بند استفاده شده است.

۴-۳-۱- انتخاب معیارهای ارزیابی کارایی برای ترکیب

گفته شد که برای ارزیابی طبقه‌بند در انتخاب ویژگی بهتر است از ترکیب چندین معیار ارزیابی کارایی استفاده شود. طبق پژوهش کو[1] و همکاران، ارتباط خطی بین دو معیار صحت و AUC کم است و به این معناست که این دو معیار از دو بعد متفاوت، کارایی طبقه‌بند را اندازه‌گیری می‌کنند. در نتیجه این دو معیار برای ارزیابی کارایی طبقه‌بند لازم بوده و غیر قابل جایگزین هستند [۳۹]. از طرفی آن‌ها در پژوهش خود به منظور مرتب‌سازی معیارها پرسشنامه‌ای را برای افراد متخصص فرستاده و پاسخ‌های آنان را میانگین گرفته‌اند. طبق نتایج به دست آمده، در معیارهای مبتنی بر کارایی، سه معیار صحت، AUC و نرخ مثبت صحیح به ترتیب بالاترین امتیاز را در طبقه‌بندهای دودویی به دست آورده‌اند. بنابراین در این پژوهش نیز از ترکیب این سه معیار استفاده شده است. اما از آنجایی که داده‌ها در حوزه پیش‌بینی خطای نرم‌افزار معمولا نامتعادل هستند، AUC عملکرد بهتری را نسبت به صحت از خود نشان می‌دهد. بیشتر ارزیابی‌ها در این حوزه نیز بر اساس این معیار انجام می‌شود و طبق پژوهش هوانگ[2] و لینگ[3]، معیار AUC به‌طور عمومی بهتر از معیار صحت است [۴۰]. بنابراین در این پژوهش اهمیت این سه معیار به‌ترتیب ۱- AUC، ۲- صحت و ۳- نرخ مثبت صحیح در نظر گرفته شده است. در نتیجه هر سه بخش انتخاب ویژگی پیشنهادی تکمیل شده و در شکل (۱۲)، مدل انتخاب ویژگی MRF آورده شده است.

[1] Kou

[2] Huang
[3] Ling

۴-۳-۲- ترکیب AUC، صحت و نرخ مثبت صحیح

روش‌های متفاوتی برای ترکیب معیارها وجود دارد. شاید ساده‌ترین روش، استفاده از تکنیک میانگین‌گیری باشد. بر این اساس که معیار نهایی یا همان میزان خوب بودن ویژگی‌ها (Goodness)، حاصل میانگین سه معیار AUC، صحت و نرخ مثبت صحیح است که در رابطه (۴) نشان داده شده است.

$$Goodness = \frac{AUC + Accuracy + TPR}{3} \quad (۴)$$

مشکل این روش در این است که این سه معیار دارای ارزش یکسانی نیستند و مجموعه ویژگی‌هایی که میانگین بالاتری دارند لزوما مجموعه ویژگی بهتری نیستند. یک روش دیگر برای ترکیب این معیارها استفاده از روش‌های MCDM[1] مانند روش جمع وزن‌دار (WSM) است که تقریبا مشابه روش میانگین وزن‌دار عمل می‌کند. دو روش WSM و میانگین وزن‌دار به ترتیب در رابطه (۵ و ۶) آورده شده است.

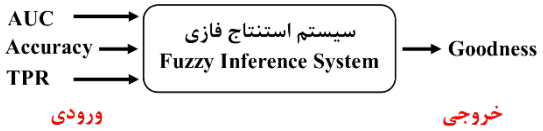
$$Goodness = (AUC \times W_1) + (Accuracy \times W_2) + (TPR \times W_3) \quad (۵)$$

$$Goodness = \frac{(AUC \times W_1) + (Accuracy \times W_2) + (TPR \times W_3)}{W_1 + W_2 + W_3} \quad (۶)$$

در حالت عمومی می‌توان وزن هر معیار را به نسبت اهمیت رتبه‌بندی آن معیار تعیین کرد و با توجه به اینکه معیار AUC بالاترین و نرخ مثبت صحیح، پایین‌ترین معیار در نظر گرفته شده‌اند، مقدار Goodness بر اساس میانگین وزن‌دار طبق رابطه (۷) محاسبه خواهد شد.

$$Goodness = \frac{(AUC \times 3) + (Accuracy \times 2) + (TPR)}{3 + 2 + 1} \quad (۷)$$

مشکل این روش در این است که نمی‌توان وزن هر معیار را به درستی به دست آورد. یعنی نمی‌توان با قطعیت بیان کرد که وزن AUC سه برابر وزن نرخ مثبت صحیح است و یا ارزش صحت، دو برابر ارزش نرخ مثبت صحیح است. اما روش ارائه شده و پیشنهادی برای ترکیب این سه معیار بر اساس سیستم استنتاج فازی است که در شکل (۱۳) نشان داده شده است. در واقع این دانش وجود دارد که معیار AUC مهم‌تر از معیار صحت است اما اینکه چقدر مهم‌تر است، یعنی وزن آن مشخص نیست. همچنین دانش دیگری وجود دارد که AUC مهم‌تر از صحت و صحت مهم‌تر از نرخ مثبت صحیح است. از این دانش در ایجاد پایگاه قوانین سیستم استنتاج فازی استفاده شده است.



شکل (۱۳). ترکیب معیارها با استفاده از سیستم استنتاج فازی

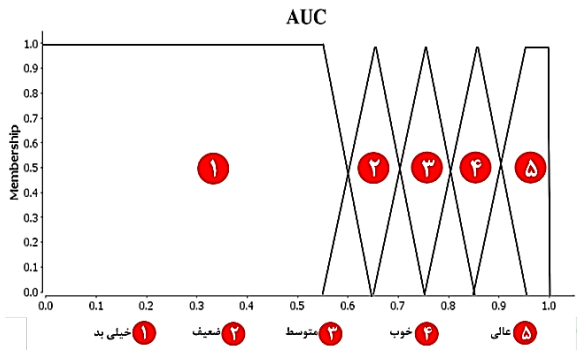
همانطور که در بخش (۲- ۶) نیز ذکر شد، این سیستم دارای سه بخش اصلی ۱- فازی‌سازی، ۲- پایگاه قوانین و موتور استنتاج و ۳- عکس فازی‌سازی است.

۴-۳-۳- فازی‌سازی و توابع عضویت

همانطور که گفته شد، سه معیار AUC، صحت و نرخ مثبت صحیح، ورودی‌های سیستم هستند و باید برای فازی‌سازی هر کدام از آن‌ها یک تابع عضویت تعریف بشود. همچنین Goodness یا همان خوب بودن نیز به عنوان خروجی در نظر گرفته شده است. از آنجایی که متداول‌ترین روش استنتاج از نوع ممدانی[2] است [۴۱]، در این پژوهش نیز از این روش استفاده شده است.

۴-۳-۳-۱- تابع عضویت AUC

در تحقیقات انجام شده، مقادیر AUC به پنج دسته تقسیم شده است. AUC بین ۰,۵ تا ۰,۶ بیانگر یک طبقه‌بندی ضعیف، بین ۰,۶ تا ۰,۷ بیانگر یک طبقه‌بندی منصفانه، بین ۰,۷ تا ۰,۸ طبقه‌بندی قابل قبول، از ۰,۸ تا ۰,۹ طبقه‌بندی عالی و بالاتر از ۰,۹ بسیار ممتاز است [۲۳, ۴۲]. بنابراین با توجه به این تقسیم‌بندی، در این پژوهش پنج متغیر زبانی ۱- خیلی بد، ۲- ضعیف، ۳- متوسط، ۴- خوب و ۵- عالی برای AUC در نظر گرفته شده است. برای معیار AUC از تابع عضویت ترکیبی مثلثی-ذوزنقه‌ای استفاده شده که در آن رأس هر مثلث، مقدار میانی در هر بازه است. مثلا رأس مثلث مربوط به مقدار AUC متوسط، مقدار میانه ۰,۷ و ۰,۸ بوده که برابر با ۰,۷۵ می‌شود. دو مقدار خیلی بد و عالی نیز به صورت ذوزنقه‌ای تعریف شده است. تابع عضویت AUC در شکل (۱۴) نشان داده شده است.



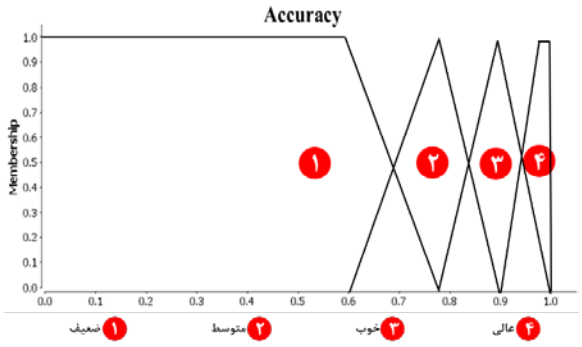
شکل (۱۴): تابع عضویت AUC

[1] Multiple Criteria Decition-Making Methods

[2] Mamdani

۴-۳-۳-۲- تابع عضویت صحت

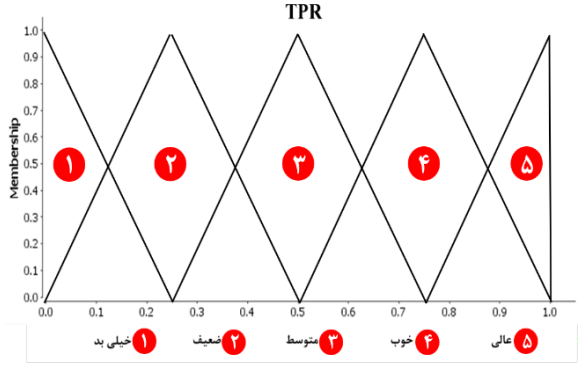
دسته‌بندی مشخصی برای مقادیر صحت وجود ندارد. پاسخ یک متخصص به این صورت است که به طور عمومی، مقدار صحت بین ۰,۵ تا ۰,۷ ضعیف ، ۰,۷ تا ۰,۸۵ متوسط، ۰,۸۵ تا ۰,۹۵ خوب و ۰,۹۵ به بالا عالی به شمار می‌رود. بنابراین چهار متغیر زبانی ۱- ضعیف، ۲- متوسط، ۳- خوب و ۴- عالی برای معیار صحت در نظر گرفته شده است. برای این معیار نیز مانند معیار AUC از تابع عضویت ترکیبی مثلثی- ذوزنقه‌ای استفاده شده که رأس هر مثلث، مقدار میانی در هر بازه است. تابع عضویت صحت در شکل (۱۵) نشان داده شده است.



شکل (۱۵): تابع عضویت صحت

۴-۳-۳-۳- تابع عضویت نرخ مثبت صحیح

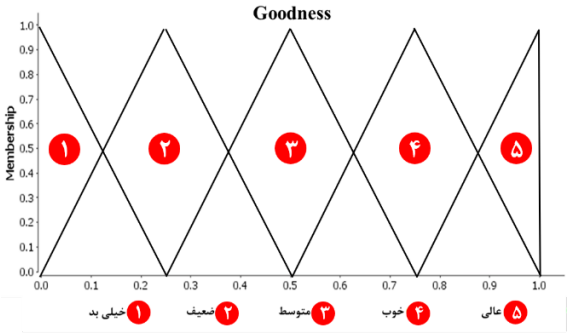
این معیار بیانگر این است که چه نسبتی از بخش‌های خطادار به درستی توسط طبقه‌بند بازگردانده شده‌اند. برخلاف معیارهای AUC و صحت که معمولا بالاتر از مقدار ۰,۵ هستند، از آنجایی که این معیار یک نسبت است، احتمال دارد که هر مقداری را بین ۰ تا ۱ داشته باشد. بر این اساس پنج متغیر زبانی ۱- خیلی بد، ۲- ضعیف، ۳- متوسط، ۴- خوب و ۵- عالی برای این معیار در نظر گرفته شده و تابع عضویت آن متشکل از پنج تابع عضویت مثلثی با اندازه‌های یکسان می‌باشد. تابع عضویت نرخ مثبت صحیح در شکل (۱۶) نشان داده شده است.



شکل (۱۶). تابع عضویت نرخ مثبت صحیح

۴-۳-۳-۴- تابع عضویت Goodness

از آنجایی که در این پژوهش، سیستم به فرم ممدانی در نظر گرفته شده است، وضعیت خروجی نیز باید به شکل فازی باشد. Goodness یا میزان خوب بودن به عنوان خروجی در نظر گرفته شده و بیانگر این است که چه مقدار ویژگی‌های انتخاب شده خوب هستند. برای این منظور از پنج متغیر زبانی ۱- خیلی بد، ۲- ضعیف، ۳- متوسط، ۴- خوب و ۵- عالی استفاده شده است. تابع عضویت آن نیز متشکل از پنج تابع عضویت مثلثی با اندازه‌های یکسان می‌باشد. تابع عضویت Goodness در شکل (۱۷) آورده شده است.



شکل (۱۷): تابع عضویت Goodness

۴-۳-۴- ایجاد پایگاه قوانین

از آنجایی که AUC و نرخ مثبت صحیح، دارای ۵ حالت و صحت دارای ۴ حالت مختلف هستند، بنابراین ۱۰۰ ترکیب مختلف ممکن است به وجود بیاید که برای هر یک از این حالت‌ها به یک قانون نیاز است. اما استفاده از افراد متخصص و خبره برای به دست آوردن این ۱۰۰ قانون دارای هزینه زیادی بوده و همچنین دسترسی به چنین افراد خبره‌ای بسیار مشکل است. بنابراین در این پژوهش برای استخراج قوانین از روشی جدید استفاده شده است. برای ایجاد قوانین در سیستم استنتاج فازی، سه نوع دانش در دسترس است. ۱- معیار AUC بیش‌ترین و معیار نرخ مثبت صحیح، کم‌ترین میزان اهمیت را در بین سه معیار ورودی دارا هستند. ۲- هر سه معیار موجود، جزء معیارهای مثبت می‌باشند یعنی هرچه مقدار بیش‌تری را داشته باشند، دارای ارزش بیش‌تری نیز هستند. مثلا معیار میانگین خطای مطلق (MAE) یک معیار منفی است و هرچه مقدار کمتری داشته باشد، ارزش بیش‌تری دارد. ۳- ارزش‌گذاری معیارها بر اساس متغیرهای زبانی و به صورت کیفی بیان شده، بنابراین مثلا AUC خوب بهتر از AUC با مقدار کیفی متوسط است. با توجه به این سه دانش، به ترتیب به معیار AUC امتیاز ۳، صحت امتیاز ۲ و نرخ مثبت صحیح، امتیاز ۱ اختصاص داده می‌شود. ذکر این نکته حائز اهمیت است که این امتیازدهی متفاوت از وزن‌دهی به

معیارهاست که در میانگین وزن‌دار به آن اشاره شد و در اینجا می‌توان از امتیازهای دیگر اما با همین ترتیب نیز استفاده کرد. سپس به بهترین متغیر زبانی در هر معیار، امتیاز ۵ را اختصاص داده و دیگر متغیرهای پایین‌تر از آن با توجه به نسبتشان ارزش‌گذاری می‌شوند. نحوه این امتیازدهی در جدول (۷) آورده شده است.

جدول (۷): امتیازدهی به معیارها و متغیرهای زبانی برای ایجاد قوانین

نرخ مثبت صحیح = ۱	صحت = ۲	AUC = ۳
عالی = ۵	عالی = ۵	عالی = ۵
خوب = ۴	خوب = ۳,۷۵	خوب = ۴
متوسط = ۳	متوسط = ۲,۵	متوسط = ۳
ضعیف = ۲	ضعیف = ۱,۲۵	ضعیف = ۲
خیلی بد = ۱		خیلی بد = ۱

سپس امتیاز هر حالت را با توجه به رابطه (۸) محاسبه کرده که در آن V_{AUC}، $V_{Accuracy}$ و V_{TPR} به‌ترتیب برابر با امتیاز متغیرهای زبانی AUC، صحت و نرخ مثبت صحیح می‌باشد.

$$Score = (3\times V_{AUC}) + (2\times V_{Accuracy}) + (1\times V_{TPR}) \quad (۸)$$

مثلا امتیاز بهترین حالت ممکن یعنی وضعیتی که هر سه معیار دارای مقدار عالی باشند طبق رابطه (۹) محاسبه می‌گردد.

$$MaximumScore = (3\times 5) + (2\times 5) + (1\times 5) = 30 \quad (۹)$$

امتیاز بدترین حالت ممکن نیز یعنی وضعیتی که دو معیار AUC و نرخ مثبت صحیح دارای مقدار خیلی بد و معیار صحت ضعیف باشد، طبق رابطه (۱۰) محاسبه می‌گردد.

$$MinimumScore = (3\times 1) + (2\times 1.25) + (1\times 1) = 6.5 \quad (۱۰)$$

در نتیجه تمامی ۹۸ حالت دیگر دارای امتیازی بین این دو مقدار کمینه و بیشینه هستند. در مرحله بعد، نرمال‌سازی امتیازها با استفاده از روش نرمال‌سازی MinMax انجام می‌شود. مقدار امتیاز نرمال شده توسط رابطه (۱۱) محاسبه می‌شود.

$$NormalizedScore = \frac{Score - Min}{Max - Min} = \frac{Score - 6.5}{23.5} \quad (۱۱)$$

بر این اساس تمامی امتیازها بین دو مقدار ۰ و ۱ قرار می‌گیرند که این دو مقدار به ترتیب متعلق به بدترین و بهترین وضعیت هستند. از آنجایی که ۵ متغیر زبانی برای Goodness در نظر گرفته شده، بازه بین ۰ و ۱ را به ۵ قسمت تقسیم کرده و هر متغیر زبانی Goodness را به یک قسمت آن نسبت می‌دهیم. این تقسیم‌بندی در جدول (۸) نشان داده شده است.

جدول (۸): اختصاص هر بازه امتیازی به یک متغیر زبانی Goodness

Goodness	بازه امتیاز
خیلی بد	۰ تا ۰,۲
ضعیف	۰,۲ تا ۰,۴
متوسط	۰,۴ تا ۰,۶
خوب	۰,۶ تا ۰,۸
عالی	۰,۸ تا ۱

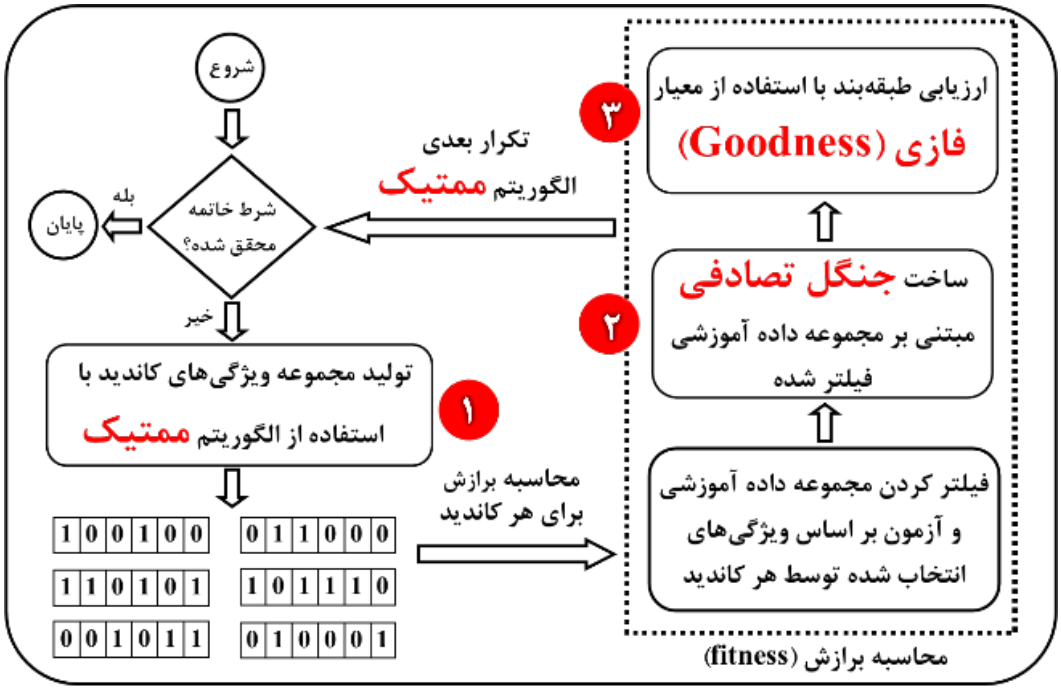
در نهایت با توجه به امتیاز نرمال‌شده هر یک از ۱۰۰ وضعیت ممکن و تقسیم‌بندی بازه امتیازات، قاعده و قانون مربوط به آن ایجاد می‌شود. به عنوان مثال یک وضعیت به این صورت است که AUC برابر با خوب، صحت برابر با ضعیف و نرخ مثبت صحیح برابر با عالی باشد. ابتدا طبق رابطه (۸) امتیاز این وضعیت به دست می‌آید که برابر با مقدار ۱۹,۵ است. سپس طبق رابطه (۱۱) این امتیاز نرمال شده که برابر با ۰,۵۵ می‌شود. همانطور که مشاهده می‌شود، امتیاز ۰,۵۵ در جدول (۸) متعلق به Goodness با مقدار متوسط است. بنابراین قانون این وضعیت به این صورت تعریف می‌شود: «اگر AUC خوب باشد و صحت ضعیف باشد و نرخ مثبت صحیح، عالی باشد آنگاه Goodness متوسط است». به جهت اختصار تنها چند نمونه از قواعد ایجاد شده در جدول (۹) آورده شده است.

جدول (۹): چندین نمونه از پایگاه قواعد ایجاد شده

امتیاز نرمال شده	Goodness	نرخ مثبت صحیح	صحت	AUC
۰,۱۴	خیلی بد	ضعیف	متوسط	خیلی بد
۰,۲۹	ضعیف	متوسط	خوب	خیلی بد
۰,۴۸	متوسط	عالی	عالی	خیلی بد
۰,۶۱	خوب	عالی	عالی	ضعیف
۰,۸۲	عالی	خوب	عالی	خوب

در پایگاه قواعد برای عمل 'and' از ضرب و برای 'or' از جمع استفاده شده است. همچنین تابع فعالساز ضرب بوده و برای عمل تجمیع نیز از بیشینه[1] استفاده شده است.

[1] Maximum



شکل (۱۸): معماری نهایی انتخاب ویژگی پیشنهادی (MRF)

۴-۳-۵- عکس فازی‌سازی

در این قسمت باید متغیر زبانی Goodness که به صورت فازی در نظر گرفته شده است، به یک مقدار کمّی تبدیل شود. در واقع خروجی نهایی کل سیستم، همین مقدار کمّی به دست آمده است که به عنوان مقدار برازش الگوریتم ممتیک در نظر گرفته می‌شود و هر چه عدد بالاتری باشد، بیانگر بهتر بودن آن دسته از ویژگی‌ها است. برای عمل عکس فازی‌سازی از روش نقطه مرکزی[1] استفاده شده است. مقدار Goodness همان معیار ارزیابی مورد استفاده در روش انتخاب ویژگی پیشنهادی است که معماری نهایی این روش در شکل (۱۸) آورده شده است.

۴-۴- تشریح روش تحقیق

روش کلی تحقیق در شکل (۱۹) نشان داده شده است. ابتدا باید بر روی مجموعه داده ورودی عمل انتخاب ویژگی انجام شود و ویژگی‌های افزونه و نامرتبط، از آن حذف شوند. فرض می‌شود که مجموعه داده مورد نظر دارای ۲۰ ویژگی نرم‌افزاری است و قرار است که روش انتخاب ویژگی ارائه شده، بر روی آن اعمال شود. ابتدا در بخش اول از الگوریتم ممتیک برای جستجو در فضای ویژگی‌ها استفاده می‌شود. به‌عنوان مثال در تکرار اول الگوریتم ممتیک، مجموعه ویژگی‌های A= {۳، ۵، ۹، ۱۰، ۱۴، ۱۷} {۱۵، ۱۴، ۱۰، ۷، ۲، ۱}=B، {۱۸، ۱۳، ۱۱، ۸}=C و {۱۸، ۱۷، ۱۵، ۱۰، ۴، ۳، ۱}=D به‌عنوان افراد جمعیت (پاسخ‌های کاندید) در نظر گرفته شده‌اند. سپس بر اساس هر یک از این چهار کاندید، مجموعه داده آموزشی و آزمون فیلتر شده و تنها ویژگی‌هایی در آن‌ها باقی می‌مانند که در مجموعه‌ی کاندید متناظر وجود داشته باشند. سپس چهار طبقه‌بند جنگل تصادفی متناظر با هر یک از این مجموعه داده‌های آموزشی فیلترشده ساخته می‌شود. در ادامه به هر یک از چهار طبقه‌بند ایجاد شده، داده آزمون فیلترشده متناظر با آن وارد شده و سه معیار AUC، صحت و نرخ مثبت صحیح طبقه‌بند، محاسبه می‌شود. سپس این سه معیار به عنوان ورودی به یک سیستم استنتاج فازی مجزا داده می‌شوند. در سیستم فازی، هر یک از این سه معیار به شکل فازی تبدیل شده و در نهایت با استفاده از پایگاه قواعد و موتور استنتاج فازی، خروجی سیستم به صورت فازی به دست آمده و با روش عکس فازی‌سازی، به یک خروجی کمّی و عددی تبدیل می‌شود. این خروجی عددی، متغیر Goodness است که یک مقدار عددی بین ۰ و ۱ را داراست که بیانگر امتیاز هر یک از آن چهار کاندید است.

[1] Centroid

در واقع مقدار Goodness همان مقدار برازش الگوریتم ممتیک است که میزان خوب بودن افراد جمعیت را بیان می‌کند. به‌عنوان مثال داریم: Goodness(A) = ۰/۵۶، Goodness(B) = ۰/۷۳، Goodness(C) = ۰/۶۱ و Goodness(D) = ۰/۶۹، در نتیجه مجموعه B دارای ویژگی‌های بهتر و مناسب‌تری است. در نهایت این فرآیند آنقدر تکرار می‌شود تا شرط پایان الگوریتم که همان حداکثر تعداد تکرار در الگوریتم ممتیک است، محقق شود. خروجی نهایی آن، همان پاسخ بهینه و یا نزدیک به مقدار بهینه بوده و مجموعه‌ای از ویژگی‌ها است که توانسته‌اند بالاترین امتیاز Goodness را به دست آوردند. در نهایت ویژگی‌های انتخاب شده که همان ویژگی‌های بهینه و مورد نظر هستند، در مجموعه داده نهایی باقی مانده و دیگر ویژگی‌ها حذف می‌شوند. سپس با استفاده از این مجموعه داده که دارای ویژگی‌های بهینه است، طبقه‌بند جنگل تصادفی نهایی ساخته می‌شود که همان مدل پیش‌بینی‌کننده خطای نرم‌افزاری مورد نظر است. از آنجایی که ویژگی‌های زائد از مجموعه داده یادگیری حذف شده است، عمل یادگیری بهتر انجام شده و به دنبال آن، مدل پیش‌بینی‌کننده دارای عملکرد بهتری نسبت به مدل اولیه است. در نهایت هم مدل پیش‌بینی کننده خطای نرم‌افزاری ارائه شده، بر اساس معیار AUC ارزیابی شده و با کارهای موجود مقایسه می‌شود.

۵- تجزیه و تحلیل داده‌ها

در این بخش نتایج پیاده‌سازی روش ارائه شده، مورد بررسی قرار گرفته و با دیگر روش‌های موجود مقایسه شده است. برای این منظور دو آزمایش مختلف در نظر گرفته شده است. در آزمایش اول، ارزیابی معیار فازی ارائه شده انجام شده تا عملکرد بهتر این معیار نسبت به دیگر معیارها مشخص شود. سپس در آزمایش دوم، مدل نهایی ارائه شده است که شامل انتخاب ویژگی به روش مذکور و ایجاد مدل پیش‌بینی‌کننده بوده و در نهایت با دیگر مدل‌ها مقایسه خواهد شد.

۵-۱- محیط آزمایش

پیاده‌سازی روش پیشنهادی با استفاده از زبان جاوا و در محیط توسعه یکپارچه Netbeans نسخه ۸,۲ انجام شده است. همچنین به منظور ساخت طبقه‌بندهای یادگیری ماشین، کتابخانه قدرتمند Weka[1] مورد استفاده قرار گرفته شده است. برای پیاده‌سازی سیستم استنتاج فازی و پایگاه قواعد مرتبط با آن نیز از کتابخانه jFuzzyLogic[2] در جاوا استفاده شده که نحوه عملکرد این کتابخانه در مقاله [۴۳] آورده شده است.

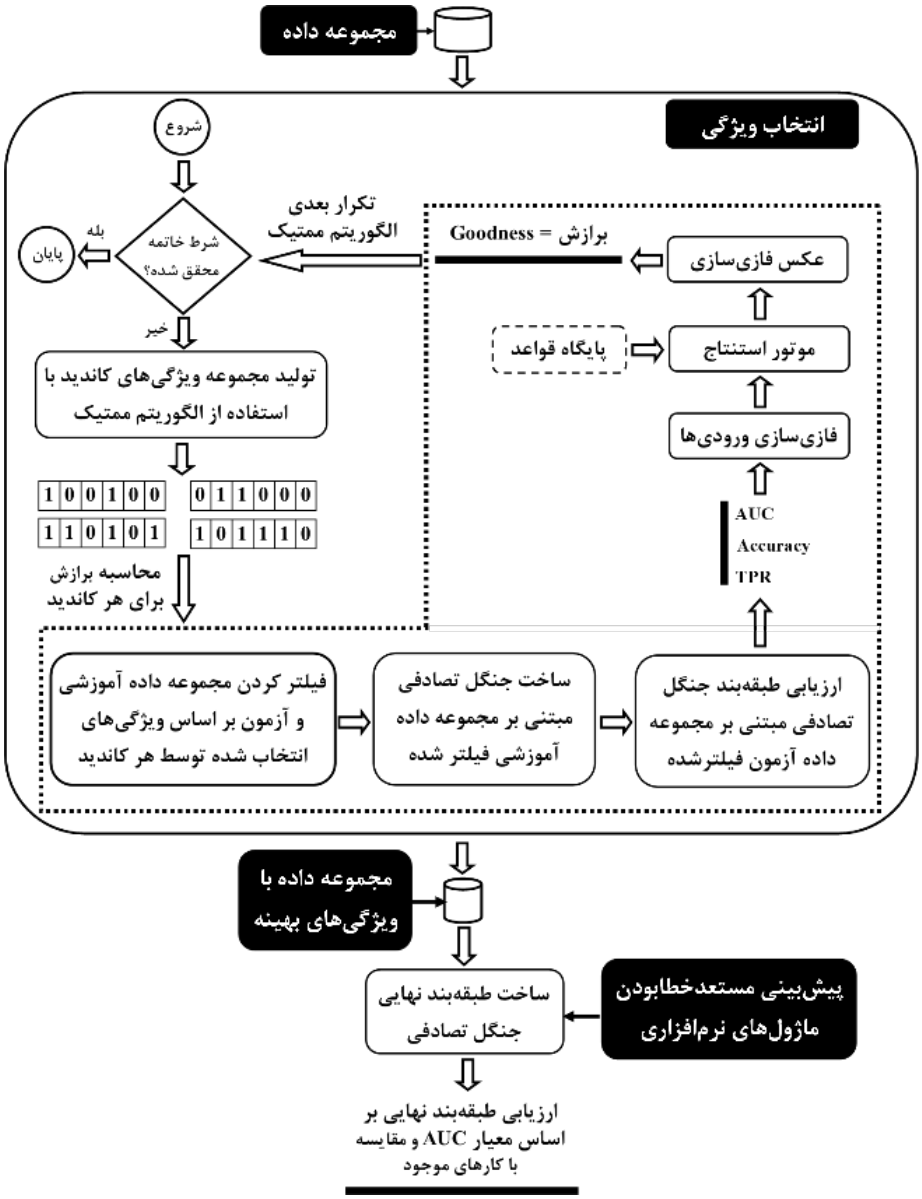
۵-۲- مجموعه داده مورد استفاده

در حوزه پیش‌بینی خطای نرم‌افزار، مجموعه داده‌های مختلفی وجود دارند که از معروف‌ترین آن‌ها می‌توان به دو مجموعه داده NASA و PROMISE اشاره کرد. همانطور که در بخش (۲-۲) بیان شد، مجموعه داده NASA از معیارهای کد و پیچیدگی تشکیل شده که در جدول (۲) به معرفی آن‌ها پرداخته شد. همچنین بیان شد که مجموعه داده PROMISE از معیارهای شیءگرایی تشکیل شده و این معیارها نیز در جدول (۳) آورده شده‌اند. برای این که هر دو مجموعه داده در بخش ارزیابی به کار گرفته شود، در هر یک از آزمایش‌ها از یک مجموعه داده استفاده شده است. از آنجایی که برای آزمایش دوم یعنی مقایسه روش پیشنهادی با کارهای موجود، پژوهش ترابیه و همکاران در نظر گرفته شده است و آن‌ها در کار خود از مجموعه داده PROMISE استفاده کرده‌اند، بنابراین به ناچار برای مقایسه روش پیشنهادی، در آزمایش دوم از مجموعه داده PROMISE استفاده شده است. در نتیجه در آزمایش اول، مجموعه داده دیگر یعنی مجموعه داده NASA مورد استفاده قرار گرفته است.

۵-۳- آزمایش اول (ارزیابی معیار فازی)

در این آزمایش به ارزیابی معیار فازی ارائه‌شده در فصل قبل پرداخته می‌شود. همانطور که گفته شد، برای ارزیابی ویژگی‌های انتخاب شده در روش‌های مبتنی بر پوشش، غالبا از معیار نرخ خطا یا همان صحت طبقه‌بند ایجاد شده بر همان ویژگی‌ها استفاده می‌شود. این معیار دارای نقاط ضعفی است و گفته شد که بهتر است از چند معیار برای این منظور استفاده شود که در نهایت سه معیار AUC، صحت و نرخ مثبت صحیح انتخاب شدند. سپس سه روش برای ترکیب این سه معیار ارائه شد که عبارتند از ۱- روش میانگین ساده، ۲- روش میانگین وزن‌دار و ۳- روش مبتنی بر سیستم استنتاج فازی. در این بخش، عملکرد این سه روش به همراه معیار صحت که در بیشتر مقالات مورد استفاده قرار گرفته و همچنین معیار AUC که معیار مهمی به شمار می‌رود، مورد مقایسه قرار می‌گیرند.

[1] https://www.cs.waikato.ac.nz/~ml/weka/
[2] http://jfuzzylogic.sourceforge.net/html/index.html



شکل (۱۹): چارت کلی روش ارائه شده

۵-۳-۱- نحوه انجام آزمایش

در این آزمایش نشان داده می‌شود که معیار فازی ارائه شده در بخش انتخاب ویژگی، به نسبت دیگر معیارها عملکرد بهتری را از خود نشان می‌دهد. برای این منظور، عملکرد شش مدل مختلف پیش‌بینی‌کننده خطای نرم‌افزاری بر روی مجموعه داده NASA با یکدیگر مقایسه خواهد شد. در مدل اول، هیچ انتخاب ویژگی انجام نشده و طبقه‌بند با استفاده از تمامی ویژگی‌ها برای طبقه‌بندی بخش‌های نرم‌افزاری ایجاد می‌شود. در پنج مدل دیگر، ابتدا عمل انتخاب ویژگی بر روی مجموعه داده انجام گرفته و سپس طبقه‌بند نهایی با استفاده از آن ویژگی‌ها ایجاد می‌شود.

تنها تفاوت این پنج مدل در بخش انتخاب ویژگی است که از پنج معیار مختلف به عنوان ارزیابی مجموعه ویژگی‌ها استفاده می‌کنند. همانطور که در جدول (۱۰) نیز مشاهده می‌شود، مدل دوم از معیار صحت که معادل با نرخ خطا است و مدل سوم از معیار AUC استفاده می‌کند. مدل چهارم از میانگین ساده $(AUC + Accuracy + TPR)/3$، مدل پنجم از میانگین وزن‌دار $[(AUC\times3)+(Accuracy\times2)+(TPR\times1)]/6$ و مدل ششم از معیار فازی ارائه شده برای ارزیابی ویژگی‌ها استفاده می‌کند. سپس نتیجه طبقه‌بندی این شش مدل بر اساس معیار AUC با یکدیگر مقایسه می‌شود. ارزیابی هر مدل با استفاده از تکنیک 5-Fold Cross-Validation انجام می‌شود. به این صورت که هر مجموعه داده به ۵ بخش تقسیم شده و در هر بار ۴ بخش آن به منظور یادگیری و ۱ بخش باقیمانده به منظور ارزیابی و تست در نظر گرفته می‌شود. این عمل ۵ بار تکرار می‌شود تا تمامی مجموعه داده مورد آزمایش قرار بگیرند و در نهایت میانگین این

۵ وضعیت به عنوان نتیجه خروجی اعلام می‌شود. از آنجایی که ممکن است توزیع داده‌ها در نتیجه نهایی تاثیرگذار باشد، کل فرآیند ارزیابی برای هر مدل ۵ بار تکرار می‌شود که در هر بار تکرار، توزیع داده‌ها به صورت تصادفی تغییر کرده و در نهایت میانگین این ۵ بار تکرار به عنوان خروجی در نظر گرفته می‌شود.

جدول (۱۰): مدل‌های مورد استفاده در آزمایش اول

بدون انتخاب ویژگی	مدل شماره ۱
استفاده از معیار صحت در انتخاب ویژگی	مدل شماره ۲
استفاده از معیار AUC در انتخاب ویژگی	مدل شماره ۳
استفاده از معیار میانگین ساده AUC، صحت و نرخ مثبت صحیح در انتخاب ویژگی	مدل شماره ۴
استفاده از معیار میانگین وزن‌دار AUC، صحت و نرخ مثبت صحیح در انتخاب ویژگی	مدل شماره ۵
استفاده از معیار فازی ارائه شده در انتخاب ویژگی	مدل شماره ۶

۵-۳-۲- پیکربندی مولفه‌های آزمایش

در شرایط مقایسه، باید پیکربندی مولفه‌های آزمایش در هر شش مدل کاملا یکسان باشد. بنابراین نحوه توزیع داده‌ها در تمامی مدل‌ها یکسان است و مقدار Seed متغیر Random که وظیفه توزیع داده‌ها را بر عهده دارد، کاملا مساوی در نظر گرفته شده است. همچنین هر شش مدل در طبقه‌بندی نهایی از طبقه‌بند جنگل تصادفی با تعداد درخت برابر با ۱۰ استفاده می‌کنند. در پنج مدلی که از انتخاب ویژگی مبتنی بر پوشش استفاده می‌کنند، از الگوریتم ممتیک برای جستجوی ویژگی‌ها و از طبقه‌بند جنگل تصادفی با تعداد ۱۰ درخت به منظور ارزیابی ویژگی‌های انتخاب شده استفاده می‌شود. در تمامی این پنج مدل، تعداد تکرار در الگوریتم ممتیک برابر با ۱۰۰ در نظر گرفته شده و همچنین اندازه جمعیت آن نیز برابر با ۱۰۰ است. در الگوریتم ممتیک عمل جستجوی محلی با استفاده از الگوریتم تپه‌نوردی انجام شده و تنها در هر ۵ بار تکرار و فقط بر روی نیمی از افراد جمعیت انجام می‌شود. همچنین نرخ جهش برابر با ۰,۰۵ تعیین شده و انتخاب افراد جمعیت بر اساس روش چرخ رولت[1] انجام می‌پذیرد.

۵-۳-۳- نتایج آزمایش

مقایسه نتایج این آزمایش بر اساس معیار AUC در جدول (۱۱) آورده شده که هر ردیف آن مربوط به یک مجموعه داده از پروژه‌های NASA و هر ستون آن نیز بیانگر هر یک از ۶ مدل معرفی شده است. در مدل اول هیچ انتخاب ویژگی انجام نشده و در مدل‌های دیگر، تنها معیار ارزیابی ویژگی‌ها متفاوت است که در جدول مذکور آورده شده است. هر خانه جدول متعلق به میانگین معیار AUC در ۵ بار تکرار آزمایش، متناسب با هر مدل و مجموعه داده مربوطه است. همانطور که مشاهده می‌شود، مدلی که از معیار فازی در انتخاب ویژگی آن استفاده شده دارای بالاترین امتیاز AUC در تمامی این شش مدل است. مدل مبتنی بر معیار AUC در جایگاه دوم قرار دارد و مطابق با انتظار، مدل‌های با معیارهای میانگین وزن‌دار، میانگین ساده و صحت به ترتیب در جایگاه‌های بعدی قرار می‌گیرند. مدلی که از انتخاب ویژگی استفاده نکرده نیز دارای پایین‌ترین امتیاز است.

هدف این آزمایش، نمایش تاثیر معیار ارزیابی مورد استفاده در انتخاب ویژگی، بر روی مدل پیش‌بینی‌کننده بود. مدل اولیه در بخش انتخاب ویژگی از معیار صحت (یا نرخ خطا) برای محاسبه برازش استفاده کرده و مدل پیشنهادی نیز از معیار فازی، در نتیجه تنها یک متغیر در مدل‌ها دچار تغییر شده که همان معیار ارزیابی مورد استفاده در تابع برازش انتخاب ویژگی است. بنابراین برای نمایش این تاثیر و انجام مقایسه آماری، آزمون آماری Wilcoxon با سطح معناداری ۰,۰۵ بر روی نتایج این دو مدل انجام شده است که در آن، مقدار p-value کوچکتر از ۰,۰۵ بیانگر تفاوت آماری قابل توجه در نتایج است. نتایج این آزمون که بر اساس معیار AUC در تمامی تکرارهای آزمایش انجام شده است، در جدول (۱۲) آورده شده و همانطور که مشخص است، کارایی مدلی که از معیار ارزیابی فازی در انتخاب ویژگی استفاده کرده بالاتر از مدلی است که از معیار صحت استفاده می‌کند.

جدول (۱۲): نتایج آزمون آماری Wilcoxon

Z	-۴/۰۱۳
P-Value	< ۰/۰۰۱

۵-۴- آزمایش دوم (ارزیابی طبقه‌بند نهایی)

در این آزمایش به ارزیابی عملکرد نهایی مدل ارائه شده پرداخته می‌شود. در این مدل ابتدا عمل انتخاب ویژگی با روش ارائه شده بر روی مجموعه داده ورودی انجام گرفته و سپس طبقه‌بند جنگل تصادفی به منظور پیش‌بینی خطا در بخش‌های نرم‌افزاری ساخته می‌شود. سپس عملکرد این طبقه‌بند با کارهای موجود مقایسه خواهد شد.

[1] Roulette Wheel

جدول (۱۱): نتایج ارزیابی آزمایش اول مبتنی بر معیار AUC

	بدون انتخاب ویژگی	Accuracy	AUC	$\frac{AUC+Acc+TPR}{3}$	$\frac{(AUC\times3)+(Acc\times2)+TPR}{6}$	روش فازی (پیشنهادی)
CM1	۰/۷۰۲۴	۰/۷۴۴۱	**۰/۷۹۴۴**	۰/۷۴۰۴	۰/۷۳۶۹	۰/۷۸۷۹
KC1	۰/۶۷۲۲	۰/۶۷۰۸	۰/۶۸۲۶	۰/۶۸۷۳	**۰/۶۹۱۶**	۰/۶۸۶۵
KC3	۰/۷۰۷۰	۰/۷۴۷۴	۰/۷۸۷۰	۰/۷۷۷۸	۰/۷۸۴۸	**۰/۷۹۳۰**
MC1	۰/۸۵۶۴	۰/۸۶۴۴	**۰/۸۸۲۲**	۰/۸۶۵۵	۰/۸۶۷۲	۰/۸۵۷۱
MC2	۰/۶۸۸۸	۰/۷۴۵۷	۰/۷۷۲۶	۰/۸۰۳۲	**۰/۸۱۱۲**	۰/۸۰۰۸
MW1	۰/۶۸۱۸	۰/۷۳۹۵	**۰/۷۸۸۰**	۰/۷۲۳۳	۰/۷۳۶۱	۰/۷۷۰۵
PC1	۰/۸۲۱۵	۰/۸۳۱۰	۰/۸۲۵۳	۰/۸۲۱۴	۰/۸۳۵۷	**۰/۸۴۴۰**
PC3	۰/۷۷۸۲	۰/۷۷۰۰	۰/۷۹۴۳	**۰/۷۹۸۸**	۰/۷۹۳۴	۰/۷۹۶۵
PC4	۰/۸۹۸۴	۰/۹۱۰۹	۰/۹۱۱۰	۰/۹۰۲۲	**۰/۹۱۷۰**	۰/۹۱۴۳
میانگین AUC	۰/۷۵۶۳	۰/۷۸۰۴	۰/۸۰۴۱	۰/۷۹۱۰	۰,۷۹۷۱ ۰/۷۹۷۱	**۰/۸۰۵۶**

۵-۴-۱- نحوه انجام آزمایش

در این آزمایش نشان داده می‌شود که مدل پیش‌بینی‌کننده خطای نرم‌افزاری ارائه شده، عملکرد بهتری را نسبت به کارهای موجود از خود نشان می‌دهد. همانطور که بیان شد، این آزمایش بر روی مجموعه داده PROMISE انجام گرفته است. ترابیه و همکاران در پژوهش [۳] کاری مشابه با روش ارائه شده انجام داده‌اند، یعنی برای بهبود پیش‌بینی خطا در نرم‌افزار، یک روش انتخاب ویژگی ارائه کرده‌اند. تفاوت روش انتخاب ویژگی پیشنهادی با کار آن‌ها در این است که در کار آن‌ها از ترکیب سه الگوریتم ژنتیک، کلونی مورچگان و ازدحام ذرات برای جستجو در فضای ویژگی‌ها استفاده شده است. همچنین برای طبقه‌بندی از شبکه عصبی و برای محاسبه برازش از معیار نرخ خطا استفاده شده است. در نهایت، آن‌ها کار خود را با دیگر کارهای موجود مقایسه کرده‌اند که نتیجه کار آن‌ها بهتر از بقیه بوده است. بنابراین فقط کافی است که روش ارائه شده با کار آن‌ها مقایسه شود. از آنجایی که در پژوهش آن‌ها و همچنین اغلب پژوهش‌ها از معیار AUC برای ارزیابی عملکرد مدل پیش‌بینی‌کننده استفاده شده است، بنابراین برای مقایسه مدل پیشنهادی نیز این معیار مورد استفاده قرار گرفته است.

۵-۴-۲- پیکربندی مولفه‌های آزمایش

در بخش انتخاب ویژگی، از الگوریتم ممتیک برای جستجوی ویژگی‌ها و از طبقه‌بند جنگل تصادفی با تعداد ۱۰ درخت به منظور ارزیابی ویژگی‌های انتخاب‌شده استفاده می‌شود. همچنین تعداد تکرار در الگوریتم ممتیک برابر با ۱۰۰ در نظر گرفته شده و اندازه جمعیت آن نیز برابر با ۱۰۰ است. در الگوریتم ممتیک عمل جستجوی محلی تنها در هر ۵ تکرار و فقط بر روی نیمی از افراد جمعیت انجام می‌شود. در این الگوریتم، نرخ جهش برابر با ۰/۰۵ تعیین شده و انتخاب افراد جمعیت بر اساس روش چرخ رولت انجام می‌پذیرد. تعداد درختان در مدل جنگل تصادفی برای طبقه‌بندی نهایی ۱۰ در نظر گرفته شده و برای متعادل‌سازی داده‌ها نیز از تکنیک نمونه‌برداری Resample استفاده شده است. نحوه ارزیابی مشابه با روش ارزیابی ترابیه و همکاران است و از روش 5-fold Cross-Validation استفاده شده است. همچنین برای ارزیابی بهتر، در هر مجموعه داده، ۱۰ مرتبه آزمایش تکرار شده است.

۵-۴-۳- نتایج آزمایش

نتایج مقایسه روش پیشنهادی با روش ترابیه و همکاران بر اساس میانگین AUC در جدول (۱۳) آورده شده است. همانطور که

مشاهده می‌شود، روش ارائه شده در مجموع عملکرد بهتری را نسبت به روش آن‌ها از خود نشان داده است.

جدول (۱۳): مقایسه روش ارائه شده با روش ترابیه و همکاران

نام پروژه	روش ترابیه و همکاران	روش پیشنهادی
ant-1.7	**۰/۸۸۴۲**	۰/۸۴۵۱
camel-1.0	**۰/۹۰۵۲**	۰/۸۸۶۸
camel-1.2	۰/۶۴۸۴	**۰/۷۳۶۶**
camel-1.4	۰/۸۷۰۷	۰/۷۷۰۸
camel-1.6	۰/۶۹۱۴	**۰/۷۵۸۴**
jedit-3.2	**۰/۹۱۷۷**	۰/۸۴۶۹
jedit-4.0	**۰/۸۷۰۰**	۰/۸۵۴۳
jedit-4.2	**۰/۸۹۱۳**	۰/۸۷۲۸
jedit-4.3	۰/۹۱۳۱	**۰/۹۴۵۱**
log4j-1.0	۰/۸۷۵۹	۰/۸۷۰۲
log4j-1.1	**۰/۹۱۴۵**	۰/۹۱۳۶
log4j-1.2	۰/۸۵۲۲	**۰/۹۴۴۵**
lucene-2.0	**۰/۸۶۷۷**	۰/۸۲۹۴
lucene-2.2	**۰/۸۲۳۶**	۰/۷۳۲۷
lucene-2.4	**۰/۸۲۵۹**	۰/۸۱۷۴
xalan-2.4	۰/۸۲۷۰	**۰/۸۲۸۹**
xalan-2.5	۰/۷۴۹۳	**۰/۷۷۰۹**
xalan-2.6	۰/۷۱۲۵	**۰/۸۵۲۶**
xalan-2.7	۰/۹۲۹۵	**۰/۹۵۴۱**
میانگین AUC	۰/۸۳۵۸	**۰/۸۴۳۷**

۵-۴-۴- تهدیدات اعتبار نتایج ارزیابی

در این بخش به مواردی اشاره خواهد شد که می‌تواند اعتبار نتایج ارزیابی را با چالش مواجه کند.

۱- برای جستجو در فضای ویژگی‌ها از الگوریتم‌های فرااکتشافی استفاده شده است. همانطور که مشخص است، این الگوریتم‌ها لزوما به پاسخ بهینه منجر نمی‌شوند، بنابراین در هر بار تکرار عمل انتخاب ویژگی، لزوما ویژگی‌های یکسانی به عنوان ویژگی‌های بهینه به دست نمی‌آیند. برای کاهش اثر این بخش، در هر آزمایش، عمل انتخاب ویژگی چندین مرتبه تکرار شده و در نهایت میانگین این تکرارها در نظر گرفته شده است.

۲- نحوه توزیع نمونه‌ها در مجموعه داده، می‌تواند در خروجی مدل تاثیرگذار باشد. بنابراین در آزمایش اول که مقایسه بین ۶ مدل مختلف انجام می‌شود، نحوه توزیع مجموعه داده در تمامی این مدل‌ها یکسان در نظر گرفته شده است. اما در آزمایش دوم که روش پیشنهادی با کار ترابیه و همکاران مقایسه شده است، چون نحوه توزیع مجموعه داده در کار آن‌ها مشخص نیست، از یک عدد ثابت به‌عنوان Seed تابع Random برای توزیع مجموعه داده استفاده شده است.

۳- عملکرد هر طبقه‌بند در معیارهای ارزیابی مختلف متفاوت است، به این معنا که یک طبقه‌بند ممکن است در یک معیار عملکرد بهتری داشته باشد و در یک معیار دیگر، عملکرد ضعیف‌تری را از خود نشان بدهد. AUC یک معیار ارزیابی معتبر بوده که حساس به توزیع داده نیست و اکثر نویسنده‌ها برای ارزیابی کار خود از آن استفاده کرده‌اند. بنابراین برای اینکه امکان مقایسه مدل پیشنهادی با کارهای مشابه فراهم شود، از این معیار برای ارزیابی مدل استفاده شده است.

۴- نحوه پیکربندی الگوریتم جستجو و طبقه‌بند، تاثیر مستقیمی بر نتایج نهایی می‌گذارد. مثلا در روش ارائه شده، تنها ۱۰۰ تکرار در الگوریتم ممتیک انجام می‌شود در حالی که در روش ترابیه و همکاران برای الگوریتم ژنتیک، ۳۰۰۰ تکرار در نظر گرفته شده است. کاملا واضح و مشخص است که هرچه این پیکربندی قدرتمندتر باشد، باعث بهبود عملکرد مدل پیش‌بینی‌کننده می‌شود.

۶- نتیجه‌گیری

یکی از روش‌های بهبود عملکرد طبقه‌بند و مدل پیش‌بینی‌کننده، انجام یک انتخاب ویژگی خوب و قدرتمند بر روی مجموعه داده است. در این پژوهش، مدلی جدید برای انتخاب ویژگی ارائه شد

که در بخش جستجو از الگوریتم ممتیک و در بخش ساخت طبقه‌بند از تکنیک جنگل تصادفی استفاده می‌کند. همچنین برای مدل انتخاب ویژگی مذکور، معیار جدیدی ارائه شد که مبتنی بر سیستم استنتاج فازی بوده و حاصل ترکیب سه معیار AUC، صحت و نرخ مثبت صحیح است. نتایج مقایسه حاکی از آن است که این معیار به نسبت معیارهای مشابه عملکرد بهتری را از خود نشان می‌دهد. در نهایت نیز مدل نهایی پیش‌بینی‌کننده خطاهای نرم‌افزاری ارائه شد که از انتخاب ویژگی مذکور استفاده کرده و برای شناسایی مستعدخطا بودن بخش‌های نرم‌افزاری نیز از روش جنگل تصادفی استفاده می‌کند. مقایسه مدل ارائه شده با کارهای موجود در این حوزه، از ایجاد بهبود روش پیشنهادی در عملکرد پیش‌بینی خطای نرم‌افزار حکایت دارد.

۷- مراجع

[1] N. Rajkumar and C. Viji, "An Efficient Software Fault Prediction Scheme to Assure Qualified Software Implementation using Improved Classification Methods, " International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 8S, 2019.

[2] J. Goyal and B. Kishan, "Progress on Machine Learning Techniques for Software Fault Prediction," International Journal of Advanced Trends in Computer Science and Engineering, vol. 8, 2019.

[3] H. Turabieh, M. Mafarja, and X. Li, "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction," Expert systems with applications, vol. 122, pp. 27-42, 2019.

[4] M. Babamir and F. Karimian, "Evaluation of Classifiers in Software Fault-Proneness Prediction," Journal of AI and Data Mining, vol. 5, 2017.

[5] Y. Jian, X. Yu, Z. Xu, and Z. Ma, "A Hybrid Feature Selection Method for Software Fault Prediction," IEICE Transactions on Information and Systems, vol. 102, no. 10, pp. 1966-1975, 2019.

[6] R. Moussa and D. Azar, "A PSO-GA approach targeting fault-prone software modules," Journal of Systems and Software, vol. 132, pp. 41-49, 2017.

[7] L. Kumar, S. K. Sripada, A. Sureka, and S. K. Rath, "Effective fault prediction model developed using least square support vector machine (LSSVM)," Journal of Systems and Software, vol. 137, pp. 686-712, 2018.

[8] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Transactions on Reliability, vol. 62, no. 2, pp. 434-443, 2013.

[9] A. Abu Zaher, R. Berretta, N. Noman, and P. Moscato, "An adaptive memetic algorithm for feature selection using proximity graphs," Computational Intelligence, vol. 35, no. 1, pp. 156-183, 2019.

[10] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: an investigation on feature selection techniques," Software: Practice and Experience, vol. 41, no. 5, pp. 579-606, 2011.

[11] X.-Y. Liu, Y. Liang, S. Wang, Z.-Y. Yang, and H.-S. Ye, "A hybrid genetic algorithm with wrapper-embedded approaches for feature selection," IEEE Access, vol. 6, pp. 22863-22874, 2018.

[12] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper–filter feature selection algorithm using a memetic framework," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 37, no. 1, pp. 70-76, 2007.

[13] E. Elakkiya and S. Selvakumar, "GAMEFEST: Genetic Algorithmic Multi Evaluation measure based FEature Selection Technique for social network spam detection," Multimedia Tools and Applications, pp. 1-33, 2019.

[14] M. Ghosh, S. Begum, R. Sarkar, D. Chakraborty, and U. Maulik, "Recursive memetic algorithm for gene selection in microarray data," Expert Systems with Applications, vol. 116, pp. 172-185, 2019.

[15] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," Pattern Recognition Letters, vol. 30, no. 5, pp. 525-534, 2009.

[16] A. Tharwat, "Classification assessment methods," Applied Computing and Informatics, 2020.

[17] S. Chatterjee and B. Maji, "A Mahalanobis distance based algorithm for assigning rank to the predicted fault prone software modules," Applied Soft Computing, vol. 70, pp. 764-772, 2018.

[18] L. A. Zadeh, "Fuzzy sets," Information and control, vol. 8, no. 3, pp. 338-353, 1965.

[19] E. Erturk and E. A. Sezer, "Iterative software fault prediction with a hybrid approach," Applied Soft Computing, vol. 49, pp. 1020-1033, 2016.

[20] Y. L. S. Rani, V. Sucharita, D. Bhattacharyya, and H.-J. Kim, "Performance evaluation of feature selection methods on large dimensional databases," International Journal of Database Theory and Application, vol. 9, no. 9, pp. 75-82, 2016.

[21] H. Faris et al., "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," Knowledge-Based Systems, vol. 154, pp. 43-67, 2018.

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

[23] R. Shatnawi, "The application of ROC analysis in threshold identification, data imbalance and metrics selection for software fault prediction," Innovations in Systems and Software Engineering, vol. 13, no. 2-3, pp. 201-217, 2017.

[24] M. M. Kabir, M. Shahjahan, and K. Murase, "A new local search based hybrid genetic algorithm for feature selection," Neurocomputing, vol. 74, no. 17, pp. 2914-2928, 2011.

[25] J. Lee, I. Yu, J. Park, and D.-W. Kim, "Memetic feature selection for multilabel text categorization using label frequency difference," Information Sciences, vol. 485, pp. 263-280, 2019.

[26] B. Chakraborty, "Feature subset selection by particle swarm optimization with fuzzy fitness function," in 2008 3rd international conference on intelligent system and knowledge engineering, vol. 1: IEEE, pp. 1038-1042, 2008.

[27] M. Bardamova, A. Konev, I. Hodashinsky, and A. Shelupanov, "A fuzzy classifier with feature selection based on the gravitational search algorithm," Symmetry, vol. 10, no. 11, p. 609, 2018.

[28] B. Chakraborty and A. Kawamura, "A new penalty-based wrapper fitness function for feature subset selection with evolutionary algorithms," Journal of Information and Telecommunication, vol. 2, no. 2, pp. 163-180, 2018.

[29] A. Kalsoom, M. Maqsood, M. A. Ghazanfar, F. Aadil, and S. Rho, "A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA)," The Journal of Supercomputing, vol. 74, no. 9, pp. 4568-4602, 2018.

[30] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Machine learning, vol. 46, no. 1-3, pp. 389-422, 2002.

[31] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," IEEE transactions on cybernetics, vol. 43, no. 6, pp. 1656-1671, 2012.

[32] L.-Y. Chuang, H.-W. Chang, C.-J. Tu, and C.-H. Yang, "Improved binary PSO for feature selection using gene expression data," Computational Biology and Chemistry, vol. 32, no. 1, pp. 29-38, 2008.

[33] P. Patchaiammal and R. Thirumalaiselvi, "Genetic Evolutionary Learning Fitness Function (GELFF) for Feature Diagnosis to Software Fault Prediction," International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 11S, 2019.

[34] P. Bermejo, J. A. Gámez, and J. M. Puerta, "Speeding up incremental wrapper feature subset selection with Naive Bayes classifier," Knowledge-Based Systems, vol. 55, pp. 140-147, 2014.

[35] C. Pascoal, M. R. Oliveira, A. Pacheco, and R. Valadas, "Theoretical evaluation of feature selection methods based on mutual information," Neurocomputing, vol. 226, pp. 168-181, 2017.

[36] M. Khonji, A. Jones, and Y. Iraqi, "An empirical evaluation for feature selection methods in phishing email classification," International Journal of Computer Systems Science & Engineering, vol. 28, no. 1, pp. 37-51, 2013.

[37] C. A. Kumar, M. Sooraj, and S. Ramakrishnan, "A comparative performance evaluation of supervised feature selection algorithms on microarray datasets," Procedia computer science, vol. 115, pp. 209-217, 2017.

[38] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," Information and Software Technology, vol. 58, pp. 388-402, 2015.

[39] G. Kou, P. Yang, Y. Peng, F. Xiao, Y. Chen, and F. E. Alsaadi, "Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods," Applied Soft Computing, vol. 86, p. 105836, 2020.

[40] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," IEEE Transactions on knowledge and Data Engineering, vol. 17, no. 3, pp. 299-310, 2005.

[41] S. Chatterjee and B. Maji, "A new fuzzy rule based algorithm for estimating software faults in early phase of development," Soft Computing, vol. 20, no. 10, pp. 4023-4035, 2016.

[42] J. N. Mandrekar, "Receiver operating characteristic curve in diagnostic test assessment," Journal of Thoracic Oncology, vol. 5, no. 9, pp. 1315-1316, 2010.

[43] P. Cingolani and J. Alcalá-Fdez, "jFuzzyLogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming," International Journal of Computational Intelligence Systems, vol. 6, no. sup1, pp. 61-75, 2013.

[44] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the nasa software defect datasets," IEEE Transactions on Software Engineering, vol. 39, no. 9, pp. 1208-1215, 2013.