«علوم و فناوری‌های پدافند نوین»

سال پانزدهم، شماره ۳، پاییز ۱۴۰۳؛ ص ۱۵۵-۱۳۷

علمی-پژوهشی

شناسایی کدهای مشکوک با استفاده از روش‌های انتخاب ویژگی مجموعه‌ای و فنون پوششی مبتنی بر شبکه عصبی

علی طلوعی‌فر[۱]، علی کریمی*[۲]، فرهاد کریمی[۳]

۱- دانشجوی کارشناسی‌ارشد ، ۲- استادیار ، ۳- پژوهشگر، دانشگاه جامع امام حسین (ع)، تهران، ایران

(دریافت: ۱۴۰۳/۰۴/۲۰، بازنگری: ۱۴۰۳/۰۶/۳۰، پذیرش: ۱۴۰۳/۰۷/۲۰، انتشار: ۱۴۰۳/۰۸/۰۱)

DOR:https://dor.isc.ac/dor/۲۰.۱۰۰۱.۱.۲۶۷۶۲۹۳۵.۱۴۰۳.۱۵.۳.۲.۳

چکیده

بازآرایی کد نرم‌افزار یکی از روش‌های مؤثر در افزایش کیفیت نرم‌افزار است که رابطه مستقیمی با کد مشکوک نرم‌افزار دارد. کد مشکوک یک نشانه سطحی است که احتمالاً نشان‌دهنده یک مشکل عمیق‌تر در برنامه است. کد مشکوک، نگهداری، توسعه و تکامل برنامه را با مشکل مواجه می‌کند. پدافند نوین مجموعه‌ای از فناوری‌ها و سیستم‌هایی است که برای مقابله با تهدیدات امنیتی مدرن و پیشرفته طراحی شده‌اند و شامل مجموعه اقداماتی نظیر طراحی امن، استفاده از الگوهای معماری مناسب و پرهیز از پیچیدگی‌های غیرضروری در کد نرم‌افزار است. بیشتر مطالعات، از نرم‌افزار متن‌باز در زبان برنامه‌نویسی جاوا استفاده کرده‌اند، درحالی‌که پروژه‌های نرم‌افزاری جدیدتر تمایل به گرایش به زبان برنامه‌نویسی پایتون دارند؛ لذا، در این مقاله، روشی بر مبنای شبکه عصبی با یک روش انتخاب ویژگی جدید شامل استفاده از روش‌های انتخاب ویژگی مجموعه‌ای و فنون پوششی جهت پیش‌بینی کدهای مشکوک نرم‌افزار در زبان پایتون ارائه شده است. کدهای مشکوک مورد اشاره در نرم‌افزار شامل متد طولانی، لیست پارامترهای طولانی، کلاس بزرگ، لیست طولانی از کلاس‌های پایه و زنجیره دامنه طولانی می‌شوند. همچنین الگوریتم ژنتیک و بهینه‌ساز گرگ خاکستری، فنون پوششی مورد نظر و بهره اطلاعاتی، امتیاز بهره اطلاعاتی و کای-مربع، سه الگوریتم مورد استفاده در بخش انتخاب ویژگی مجموعه‌ای هستند. هدف نهایی و اصلی این مقاله که بهبود کیفیت نرم‌افزار باپیش‌بینی زودهنگام کدهای مشکوک با استفاده از یک روش انتخاب ویژگی مطلوب در کد منبع برنامه با زبان پایتون است، به کمک روش پیشنهادی تحقق پیدا کرده و بهبود عملکرد ۱ تا ۷ درصد به‌دست‌آمده است.

کلیدواژه‌ها: مهندسی نرم‌افزار، بهبود کیفیت نرم‌افزار، کد مشکوک، طبقه‌بندی، شبکه عصبی، انتخاب ویژگی.

Code Smells Detection Using Ensemble Feature Selection Methods and Wrapper Techniques Based on Neural Network

A. Tolooeifar, A. Karimi, F. Karimi

Imam Hossein University, Tehran, Iran

(Received:۲۰۲٤/۰۷/۱۰, Revised: ۲۰۲٤/۰۹/۲۰, Accepted: ۲۰۲٤/۱۰/۱۱, Published: ۲۰۲٤/۱۰/۲۲)

Abstract

Software code refactoring is one of the effective methods to enhance software quality, which has a direct relationship with the code smells of the software. Code smell is a superficial symptom that may indicate a deeper problem in the application. Code smell hinders the path of maintenance, development and evolution of the program. Advanced defense is a set of technologies and systems designed to counter modern and advanced security threats. It encompasses a set of measures such as secure design, the use of appropriate architectural patterns, and avoiding unnecessary complexities in software code. Most studies have utilized open-source software in the Java programming language, whereas newer and more modern software projects tend to lean towards Python. Therefore, in this paper, a neural network-based approach with a new feature selection method including the use of ensemble feature selection methods and wrapper techniques, has been presented to predict software code smells in the Python programming language. The code smells mentioned in the software include long method, long parameter list, large class, long base class list and long scope chaining. Also, the genetic algorithm and the gray wolf optimizer are the desired wrapper techniques, and information gain, information gain ratio and chi-square are the three algorithms used in the ensemble feature selection. The final and main goal of this paper, which is to improve software quality by early prediction of code smells using a desirable feature selection method in the source code of the program with Python language, has been realized with the help of the proposed method and performance improvement of ۱ to ۷ percent has been obtained.

Keywords: Software engineering, Software quality improvement, Code smell, Classification, Neural network, Feature selection.

*Corresponding Author E-mail: mkarticles۹۰@gmail.com

۱. مقدمه

توسعه نرم‌افزارهای با کیفیت بالا در حوزه مهندسی نرم‌افزار از اهمیت زیادی برخوردار است. امروزه سامانه‌های نرم‌افزاری از نظر اندازه و پیچیدگی در حال رشد هستند؛ در نتیجه حفظ کیفیت مطلوب محصول یکی از مهمترین مشکلات پیش‌روی صنعت نرم‌افزار است [۱]. وقتی صحبت از نگهداری نرم‌افزارهای بزرگ و پیچیده می‌شود، کدهای مشکوک[۱] یک چالش واقعی است [۲]. آنها ناهنجاری‌های طراحی هستند که نرم‌افزار را پیچیده، مدیریت و درک آن را دشوار می‌کنند و منجر به هزینه‌های مالی برای توسعه و نگهداشت نرم‌افزار می‌شوند [۳].

به‌طورکلی و بنا به گفته فاولر [۴]، «کد مشکوک یک نشانه سطحی است که معمولاً مربوط به یک مشکل عمیق‌تر در سیستم است» و آنها مشخصه‌هایی در کد منبع نرم‌افزار هستند. در حالی که کد مشکوک ممکن است همیشه نشان‌دهنده یک مشکل جدی خاص نباشد، اغلب منجر به کشف این مسائل می‌شود [۵]. ابزارهای مختلف شناسایی کدهای مشکوک بر اساس بصری‌سازی، رویکرد مبتنی بر ماشین و ارزیابی بر اساس متریک‌های مختلف وجود دارند و این روش‌ها به صورت دستی، خودکار و نیمه‌خودکار هستند [۲]. بر خلاف اشکالات در کد منبع[۲]، یک برنامه با کد مشکوک می‌تواند به‌راحتی اجرا شود. این کدها، اگرچه تأثیر کوتاه‌مدتی بر اجرای برنامه ندارند، اما تأثیر طولانی‌مدت کد مشکوک باعث کاهش کیفیت نرم‌افزار شده و درک و نگهداری آن را دشوار می‌کند [۶].

اولبریش و همکاران [۷] نشان دادند که تشخیص دستی کدهای مشکوک برای کاربردهای صنعتی بزرگ غیرعملی است. بنابراین، اکثر محققان رویکردهای خودکار را برای شناسایی سریع و دقیق کدهای مشکوک ترجیح می‌دهند. ابزارهای تشخیص خودکار اولیه، عمدتاً بر معیارهای کد و قوانین اکتشافی[۳] متکی هستند [۸]. برای شناسایی کدهای مشکوک، معمولاً معیارها به صورت دستی انتخاب شده و آستانه‌ای برای قضاوت در مورد اینکه آیا ساختارهای کد می‌توانند این معیارها و قوانین را برآورده کنند یا خیر، تعیین می‌شود. از زمانی که اولین ابزار تشخیص خودکار کدهای مشکوک معرفی شد، بسیاری از ابزارهای معروف (مانند DECOR، JDedorant و غیره) برای بهبود دقت[۴] تشخیص این کدها پیشنهاد شده‌اند [۶]. محققان برای ارائه یک نمای کلی از رویکردهای مختلف و تحقیقات اخیر در مورد تشخیص کدهای مشکوک، آنها را از جنبه‌های مختلف بررسی می‌کنند. به‌عنوان مثال، ژانگ و ژو [۹] تاثیر کدهای مشکوک را بر تکامل نرم‌افزار تحلیل کردند و آنها را از دیدگاه تجربی مورد مطالعه قرار دادند. تیان و همکاران [۱۰] تفاوت در تأکید بر کدهای مشکوک بین صنعت و دانشگاه و تأثیر آن را بررسی کردند. عظیم و همکاران [۱۱] و کارام و همکاران [۱۲] به‌طور سیستماتیک رویکردهای مختلف در تشخیص کدهای مشکوک را تجزیه‌وتحلیل کردند. بیشتر مطالعات از نرم‌افزار متن‌باز در زبان برنامه‌نویسی جاوا استفاده کرده‌اند [۱۳]، در حالی که پروژه‌های نرم‌افزاری جدیدتر و مدرن‌تر تمایل به گرایش به پایتون دارند. پروژه‌های نرم‌افزاری در پایتون نیز از مشکلات نگهداری رنج می‌برند.

به این ترتیب، تشخیص کد مشکوک برای زبان برنامه‌نویسی پایتون، نقش مهمی در کاهش مشکلات ایفا می‌کند [۱۴]. از این‌رو، یک رویکرد پیش‌بینی کد مشکوک در زبان برنامه‌نویسی پایتون، بر اساس الگوریتم‌های انتخاب ویژگی[۵] و فنون یادگیری ماشین[۶] ارائه می‌شود. در روش پیشنهادی، از روش‌های انتخاب ویژگی مجموعه‌ای (شامل بهره اطلاعاتی[۷]، امتیاز بهره اطلاعاتی[۸] و کای-مربع[۹]) و فنون انتخاب ویژگی پوششی (شامل الگوریتم ژنتیک[۱۰] و بهینه‌ساز گرگ خاکستری[۱۱]) استفاده شده است. هدف از این مرحله، انتخاب مرتبط‌ترین ویژگی‌ها در هر مجموعه داده (تولیدشده توسط چن و همکاران [۱۵] در سال ۲۰۱۸ و بهبود داده‌شده توسط واتاناپاکورن و همکاران [۱۴] در سال ۲۰۲۲)، برای افزایش کارآمدی مدل یادگیری ماشین است.

همچنین از نظریه آشوب[۱۲]، جهت تولید جمعیت اولیه در الگوریتم ژنتیک و بهینه‌ساز گرگ خاکستری استفاده خواهد شد. کدهای مشکوک موردبررسی در این روش عبارت است از: متد طولانی[۱۳]، لیست پارامترهای طولانی[۱۴]، کلاس بزرگ[۱۵]، لیست طولانی از کلاس‌های پایه[۱۶] و زنجیره دامنه طولانی[۱۷].

دیگر بخش مقاله به شرح زیر تنظیم شده است. بخش دوم شامل مروری بر کارهای تحقیقاتی انجام‌شده توسط دیگر پژوهشگران است. بخش سوم به بررسی مفاهیم پایه‌ای مورد نیاز برای درک بهتر خوانندگان پرداخته شده است. بخش چهارم مقاله شامل کلیات روش تحقیق و مدل پیشنهادی و تشریح جزءبه‌جزء مراحل است. در بخش پنجم، ارزیابی روش پیشنهادی و مقایسه نتایج با مقاله مرجع، ارائه شده است. در نهایت، بخش ششم شامل نتیجه‌گیری حاصل از مفاهیم بیان‌شده در این مقاله و کارهای آینده است.

۱. Code Smells
۲. Source Code
۳. Heuristic Rules
٤. Accuracy

٥. Feature Selection (FS)
٦. Machine Learning (ML) Techniques
٧. Information Gain (IG)
٨. Information Gain Ratio (IGR)
٩. Chi-Square
١٠. Genetic Algorithm (GA)
١١. Grey Wolf Optimizer (GWO)
١٢. Chaos Theory
١٣. Long Method
١٤. Long Parameter List
١٥. Large Class
١٦. Long Base Class List
١٧. Long Scope Chaining

۲. پیشینه تحقیق

ژانگ و همکاران [۶] در سال ۲۰۲۴، ۸۶ مقاله تشخیص کدهای مشکوک بر اساس یادگیری ماشین نظارت‌شده از ژانویه ۲۰۱۰ تا آوریل ۲۰۲۳ را جمع‌آوری کردند. در این مقاله، آنها در مجموع ۷ سوال تحقیقاتی را از جنبه‌های مختلف مانند ساخت مجموعه‌داده‌ها، پیش‌پردازش داده‌ها، انتخاب ویژگی، آموزش مدل و غیره به‌صورت تجربی ارزیابی کردند. آنها نتیجه گرفتند که کارهای موجود با مشکلاتی مانند عدم‌تعادل نمونه‌ها، تمرکز متفاوت بر روی انواع کدهای مشکوک و انتخاب ویژگی محدود روبه‌رو هستند.

عبدو و درویش [۱۶] در سال ۲۰۲۴، در پژوهش خود بر اندازه‌گیری طبقه‌بندی شدت کدهای مشکوک بسته به چندین مدل یادگیری ماشین مانند مدل‌های رگرسیون[۱]، مدل‌های چندجمله‌ای[۲] و مدل‌های طبقه‌بندی ترتیبی[۳] تمرکز کردند. آنها از از الگوریتم توضیح‌های آگنوستیک مدل قابل تفسیر محلی[۴] بیشتر بیشتر برای توضیح پیش‌بینی‌ها و قابلیت تفسیر مدل یادگیری ماشین استفاده کردند. همچنین، آنها قوانین پیش‌بینی تولیدشده توسط الگوریتم نظریه تشدید انطباقی تصویری[۵] را استخراج کردند کردند تا اثربخشی استفاده از معیارهای نرم‌افزاری برای پیش‌بینی کدهای مشکوک را بررسی کنند. نتایج آزمایش‌ها نشان داد که دقت مدل طبقه‌بندی شدت نسبت به خط‌پایه افزایش یافته و همبستگی رتبه‌بندی بین مدل پیش‌بینی‌شده و واقعی با استفاده از معیار همبستگی اسپیرمن[۶] به ۰/۹۷ - ۰/۹۲ می‌رسد.

سندوکا و الجمان [۱۷] در سال ۲۰۲۳، مجموعه‌داده کدهای مشکوک زبان پایتون را برای کدهای مشکوک کلاس بزرگ و متد طولانی تولید کردند. مجموعه‌داده ساخته‌شده شامل ۱۰۰۰ نمونه برای هر کد مشکوک، با ۱۸ ویژگی استخراج‌شده از کد منبع بود. همچنین، عملکرد تشخیص شش مدل یادگیری ماشین را به عنوان خطوط‌پایه در تشخیص کد مشکوک زبان پایتون بررسی کردند. خطوط‌پایه بر اساس معیارهای دقت و ضریب همبستگی متیوز (MCC) ارزیابی شدند. نتایج نشان‌دهنده برتری گروه جنگل تصادفی در شناسایی کد مشکوک کلاس بزرگ در پایتون با دستیابی به بالاترین عملکرد تشخیص نرخ MCC، ۰/۷۷ بود، در حالی که درخت تصمیم با دستیابی به بالاترین نرخ MCC برابر با ۰/۸۹ بهترین مدل در تشخیص کد مشکوک متد طولانی پایتون بود.

واتاناپاکورن و همکاران [۱۴] در سال ۲۰۲۲، تحقیق خود را در رابطه با تشخیص کدهای مشکوک متد طولانی، لیست پارامترهای طولانی، کلاس بزرگ، لیست طولانی از کلاس‌های پایه و زنجیره دامنه طولانی در هر دو سطح کلاس و متد، با استفاده از انتخاب ویژگی مبتنی بر همبستگی[۷] و انتخاب گام‌به‌گام رگرسیون رگرسیون لجستیک به جلو[۸] (شرطی) برای بهبود عملکرد مدل مبتنی بر یادگیری ماشین برای برنامه‌های زبان پایتون انجام دادند. آنها هشت مدل یادگیری ماشین را با یک مجموعه‌داده مبتنی بر ۱۱۵ پروژه منبع باز زبان پایتون، ۳۹ معیار نرم‌افزار در سطح متد و ۲۲ معیار نرم‌افزار در سطح عملکرد، آموزش دادند. نتایج تحقیق آنها نشان داد که روش یادگیری ماشین هنگام شناسایی متد طولانی و لیست طولانی از کلاس‌های پایه، دقت ۹۹٪/۷۲ را به دست آورد.

جین و ساها [۱۸] در سال ۲۰۲۲، تحقیق خود را در رابطه با شناسایی کدهای مشکوک کلاس داده، کلاس خدا، ویژگی حسادت، متد طولانی با استفاده از سه الگوریتم انتخاب ویژگی مبتنی بر فیلتر بهره اطلاعاتی، امتیاز فیشر و ROC AUC انجام دادند. نتایج تحقیق آن‌ها نشان داد که از بین الگوریتم‌های انتخاب ویژگی مورد استفاده، الگوریتم بهره اطلاعاتی بهترین نتایج را ارائه داده است.

السغایر و همکاران [۱۹] در سال ۲۰۲۱، یک رویکرد با ادغام الگوریتم ژنتیک با طبقه‌بندی ماشین بردار پشتیبان[۹] و الگوریتم بهینه‌سازی وال[۱۰] برای پیش‌بینی کدهای مشکوک ارائه دادند. رویکرد توسعه‌یافته برای ۲۴ مجموعه‌داده (۱۲ پروژه NASA MDP و ۱۲ پروژه منبع باز جاوا) که در آن NASA MDP به‌عنوان یک مجموعه‌داده در مقیاس بزرگ و پروژه‌های منبع باز جاوا به‌عنوان یک مجموعه‌داده در مقیاس کوچک در نظر گرفته شدند. نتایج نشان داد که ترکیب الگوریتم ژنتیک با الگوریتم ماشین بردار پشتیبان و الگوریتم بهینه‌سازی وال عملکرد فرآیند پیش‌بینی کدهای مشکوک نرم‌افزار را هنگامی که در مجموعه‌داده‌های مقیاس بزرگ و کوچک اعمال شد، بهبود یافت و بر محدودیت‌های موجود در مطالعات قبلی غلبه کرد.

عظیم و همکاران [۱۱] در سال ۲۰۱۹، در مقاله خود مروری بر ادبیات سیستماتیک در فنون یادگیری ماشین برای تشخیص کدهای مشکوک ارائه کردند. آن‌ها مقالات منتشرشده بین سال‌های ۲۰۰۰ و ۲۰۱۷ را در نظر گرفتند. با شروع از مجموعه اولیه ۲۴۵۶ مقاله، متوجه شدند که ۱۵ مورد از آن‌ها در واقع رویکردهای یادگیری ماشین را اتخاذ کرده‌اند. آن‌ها، آن مقالات را تحت چهار دیدگاه مختلف مورد مطالعه قرار دادند: کد مشکوک در

۱. Regression Models
۲. Multinominal Models
۳. Ordinal Classification Models
۴. Local Interpretable Model Agnostic Explanations (LIME) Algorithm
۵. Projective Adaptive Resonance Theory (PART) Algorithm
۶. Spearman's Correlation Measure
۷. Correlation-Based Feature Selection (CFS)
۸. Logistic Regression-Forward Stepwise (LRFS)
۹. Support Vector Machine
۱۰. Whale Optimization Algorithm (WOA)

نظر گرفته شده، راه اندازی رویکردهای یادگیری ماشین، طراحی استراتژی‌های ارزیابی و یک فراتحلیل در مورد عملکرد به‌دست‌آمده توسط مدل‌های پیشنهادی.

فونتانا و همکاران [۲۰] در سال ۲۰۱۶، مدل پیش‌بینی را با استفاده از ۳۲ رویکرد یادگیری ماشینی نظارت‌شده (۱۶ الگوریتم اصلی و فنون تقویت آنها) برای تشخیص کد مشکوک ساختند. آنها بر روی چهار نوع کد مشکوک (کلاس بزرگ، کلاس داده، کلاس طولانی و ویژگی حسادت) تمرکز کرده و آزمایش‌های گسترده‌ای روی ۷۴ سیستم برای انتخاب بهترین الگوریتم‌ها با بهترین پارامترها برای تشخیص کدهای مشکوک انجام دادند. نتایج نشان داد که الگوریتم‌های آزمایش‌شده مانند J٤٨، بیز ساده و جنگل تصادفی عملکرد بالایی را به دست آوردند که به محدوده ۹۶/۶۴٪ تا ۹۹/۰۲٪ دقت بدون توجه به انواع کد مشکوک رسید.

۳. مرور ادبیات تحقیق

در زمینهٔ پیش‌بینی کدهای مشکوک مفاهیمی وجود دارد که برای درک بهتر از تحقیق حاضر ابتدا به بررسی این مفاهیم می‌پردازیم.

۳-۱. انتخاب ویژگی

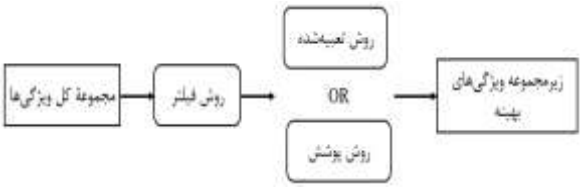
با پیشرفت سریع فناوری‌های رایانه‌ای و پایگاه‌داده، اکنون مجموعه‌داده‌هایی با صدها و هزاران متغیر یا ویژگی در پیش‌بینی الگو، داده‌کاوی و یادگیری ماشین وجود دارند [۲۱]. از طرفی، کارایی یک مدل پیش‌بینی کننده، بسیار وابسته به کیفیت مجموعه‌داده است. در مباحث و مسائل مربوط به پیش‌بینی، یک مجموعه‌داده معمولاً شامل تعداد زیادی از معیارهای نرم‌افزاری (ویژگی) است که ممکن است بسیاری از آن‌ها نامرتبط باشند. ویژگی‌های نامرتبط، اطلاعات مفیدی را برای ساخت مدل ارائه نمی‌دهند. بنابراین، ممکن است که این دسته از ویژگی‌ها منجر به کاهش کارایی و دقت طبقه‌بند شوند [۲۲]. نتایج تحقیقاتی ژاو و همکارانش [۲۳] نشان می‌دهد که در مورد مطالعاتی آن‌ها، حذف ۸۵٪ از ویژگی‌ها و معیارهای نرم‌افزاری تأثیری بر دقت پیش‌بینی نداشته و حتی در برخی موارد، کارایی نیز بهبود یافته است [۲۴].

انتخاب ویژگی یک روش پیش‌پردازش برای کاهش تعداد ویژگی‌ها است که با کنار گذاشتن ویژگی‌های نامربوط و تکراری و حفظ مرتبط‌ترین ویژگی‌ها، انجام می‌شود. اِعمال انتخاب ویژگی باعث افزایش قابلیت اطمینان و بهبود عملکرد مدل‌های ML می‌شود [۱۷]. همچنین، به‌کارگیری انتخاب ویژگی، یک روش منطقی برای مدیریت داده‌هایی با تعداد زیادی ویژگی است [۲۵].

سه روش اصلی انتخاب ویژگی به منظور تولید زیرمجموعه‌های فضای ویژگی و کمک به عملکرد بهینه مدل عبارت‌اند از: فیلتر[۱]، پوشش[۲] و روش‌های تعبیه‌شده[۳]. هر روش، کاربردها و رویکرد منحصر به فرد خود را در تعامل با ویژگی‌ها دارد. با این حال، دو روش جدید نیز اضافه شده است: روش‌های ترکیبی[۴] و مجموعه‌ای[۵] [۲۶]. بسیاری از محققان از این روش‌های جدید در مدل‌های طبقه‌بندی خود برای تولید روش‌های انتخاب ویژگی جدید استفاده می‌کنند. هر یک از این پنج روش، ویژگی‌های متمایز و خاص خود را دارند [۲۷]. نقاط قوت و ضعف خاص هر دسته روش‌های مذکور، به این معنی است که آنها برای موارد استفاده خاص، مناسب‌تر هستند [۲۸-۳۱]. مقایسه سه فن اصلی در جدول (۱) نشان داده شده است [۳۲].

۳-۱-۱. روش ترکیبی

در سال‌های اخیر، این روش یکی از روش‌های پرکاربرد مورد استفاده محققان برای اعمال فن انتخاب ویژگی است. این روش دو یا چند الگوریتم انتخاب ویژگی را به‌صورت متوالی و پیاپی از فنون متفاوت، با هم ترکیب می‌کند تا از مزایای الگوریتم‌های انتخاب ویژگی مختلف برای رسیدن به نتایج بهینه استفاده کند. هدف آن، استفاده از فنون مختلف برای غلبه بر معایب الگوریتم‌های تکی و کاهش پیچیدگی انتخاب ویژگی‌های مرتبط از مجموعه‌داده با کاهش زمان انتخاب است. این روش‌ها معمولاً به دقت بالا و همچنین پیچیدگی و کارایی محاسباتی بالاتری در مقایسه با روش‌های فیلتر دست پیدا می‌کنند [۲۷, ۳۳]. در شکل (۱) نحوه عملکرد روش‌های ترکیبی، نشان داده شده است که در این روش‌ها، ویژگی‌ها ابتدا از طریق استفاده از یک روش فیلتر کاهش می‌یابند. سپس مجموعه ویژگی‌های کاهش‌یافته از طریق یک روش پوشش یا تعبیه‌شده عبور می‌کنند تا زیرمجموعه ویژگی‌های بهینه حاصل شود [۳۲].



شکل ۱. نحوه عملکرد الگوریتم‌های انتخاب ویژگی ترکیبی

۳-۱-۲. روش مجموعه‌ای

اخیراً روش‌های انتخاب ویژگی مبتنی بر یادگیری مجموعه‌ای[۶] یا روش‌های انتخاب ویژگی مجموعه‌ای توسعه یافته‌اند که از ایده یادگیری مجموعه‌ای برای غلبه بر مشکل بهینه محلی بالقوه الگوریتم‌های مجزا بهره می‌برند. هدف از انتخاب ویژگی مبتنی بر یادگیری مجموعه‌ای، تولید زیرمجموعه‌های متنوع ویژگی‌ها از یک مجموعه‌داده آموزشی و سپس تجمیع آنها در یک خروجی نهایی است. برتری انتخاب ویژگی مبتنی بر یادگیری مجموعه‌ای نسبت

۱. Filters

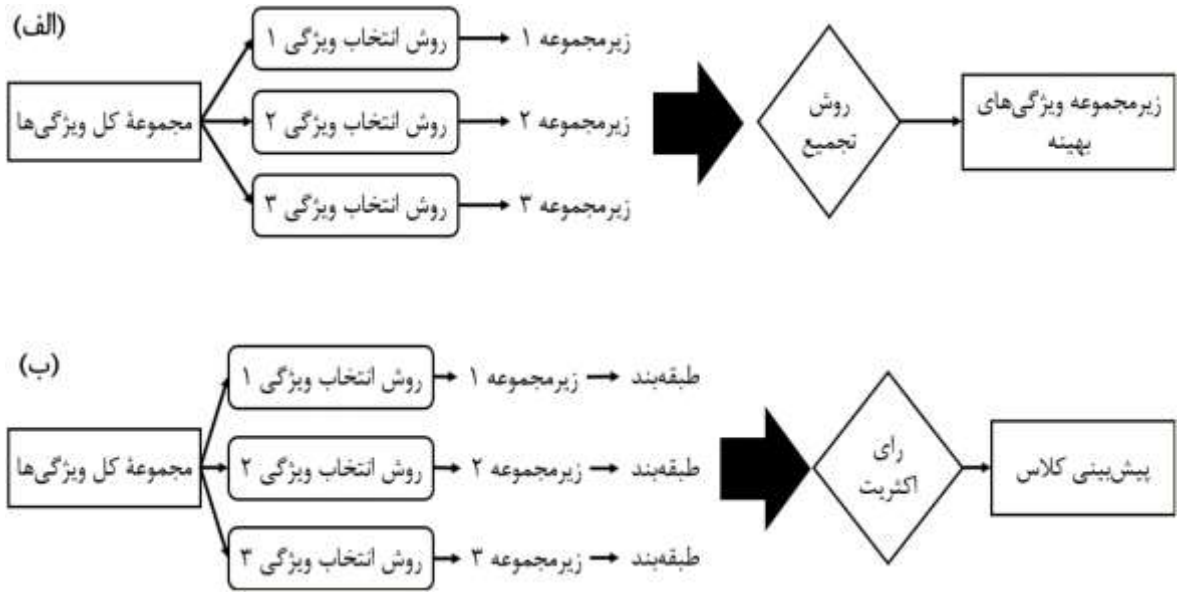
۲. Wrappers
۳. Embedded Methods
٤. Hybrid
٥. Ensemble
٦. Ensemble Learning Based Feature Selection

به روش‌های انتخاب ویژگی تکی، در مطالعات بسیاری در شکل (۲) نحوه عملکرد الگوریتم‌های انتخاب ویژگی مجموعه‌ای نشان داده شده است. در قسمت (الف)، خروجی‌های چندین روش انتخاب ویژگی برای به دست آوردن ویژگی‌های انتخابی نهایی تجمیع می‌شوند. در قسمت (ب)، تصویر کلی سیستم رای اکثریت نشان‌داده‌شده است [۳۴].

که در آن از زیرمجموعه‌های ویژگی تولیدشده مختلف برای آموزش و آزمایش یک طبقه‌بند خاص استفاده می‌شود. خروجی نهایی، کلاسی است که توسط اکثر طبقه‌بندها پیش‌بینی شده است.

جدول ۱- مقایسه سه دسته اصلی انتخاب ویژگی [۳۲]

فنون انتخاب ویژگی	نقاط قوت	نقاط ضعف	مثال
فیلتر - یک‌متغیره	- سریع و مقیاس‌پذیر - مستقل از طبقه‌بند - کاهش خطر بیش برازش	- مدل‌نشدن وابستگی‌های ویژگی‌ها - مدل‌نشدن تعامل با طبقه‌بند	- X۲ / آزمون کای-مربع - آزمون دقیق فیشر[۱] - همبستگی پیرسون[۲] - بهره اطلاعاتی
فیلتر - چندمتغیره	- توانایی مدل‌کردن وابستگی‌های ویژگی‌ها - مستقل از طبقه‌بند - ریسک کمتر برای بیش برازش	- کندتر و به‌اندازه فیلترهای تک‌متغیره مقیاس‌پذیر نیست - مدل‌نشدن تعامل با طبقه‌بند	- فیلتر مبتنی بر همبستگی سریع[۳] - حداقل - افزونگی - حداکثر - ارتباط[۴] - الگوریتم‌های مبتنی بر Relief
پوشش	- مدل‌کردن وابستگی‌های ویژگی‌ها - عملکرد بهتر از فن فیلتر - مدل‌کردن تعامل با طبقه‌بند	- کندتر از روش‌های فیلتر و تعبیه‌شده - مستعدتر به بیش برازش - وابستگی ویژگی‌های انتخاب‌شده به طبقه‌بند	- انتخاب متوالی به جلو و به عقب[۵] - تپه‌نوردی تصادفی[۶] - الگوریتم ژنتیک - الگوریتم بهینه‌ساز گرگ خاکستری
تعبیه‌شده	- مدل‌کردن وابستگی‌های ویژگی‌ها - سریع‌تر از فن پوشش - مدل‌کردن تعامل با طبقه‌بند	- کندتر از روش‌های فیلتر - وابستگی ویژگی‌های انتخاب‌شده به طبقه‌بند	- جنگل تصادفی[۷] - رگرسیون لاسو (L۱) یا رگرسیون شبکه الاستیک[۸]



۱. Fisher 'S Exact Test
۲. Pearson Correlation
۳. Fast Correlation-Based Filter
٤. Minimal-Redundancy-Maximal-Relevance
٥. Sequential Forward and Backward Selection
٦. Randomized Hill Climbing
۷. Random Forest (RF)
۸. Lasso (L۱) or Elastic Net Regression

شکل ۲. نحوه عملکرد الگوریتم‌های انتخاب ویژگی مجموعه‌ای

۳-۲. الگوریتم‌های فرااکتشافی

الگوریتم‌های فرااکتشافی[۱] سطح بالاتری از الگوریتم‌های اکتشافی هستند که برای یافتن روش‌های مسائل بهینه‌سازی استفاده می‌شوند [۳۵]. این الگوریتم‌ها فنون جستجوی همه منظوره‌ای هستند که معمولاً مستقل از مسئله هستند [۳۶]. الگوریتم‌های فرااکتشافی دو فاز اصلی به نام‌های اکتشاف و بهره‌برداری دارند. در فاز اکتشاف، روش‌های مختلف به منظور اکتشاف فضای جستجو برای یافتن بهینهٔ سراسری یافت می‌شوند. در فاز بهره‌برداری، جستجوی محلی با استفاده از اطلاعات بهترین روش‌های یافت‌شدهٔ اخیر انجام می‌شود [۳۷].

۳-۲-۱. الگوریتم ژنتیک

در سال ۱۹۶۰، هالند و همکارانش الگوریتم ژنتیک (GA) را بر اساس ایده کلی که از مفهوم نظریه تکاملی الهام گرفته شده بود، پیشنهاد کردند [۳۸]. GA یک الگوریتم تکاملی است که افراد بهینه‌شده را از مجموعه‌ای از روش‌های اولیه تولید می‌کند. عملیات اصلی الگوریتم ژنتیک از سه عامل ژنتیکی اصلی تشکیل می‌شود که شامل انتخاب[۲]، ترکیب[۳] و جهش[۴] هستند [۳۹]. شکل شکل (۳) شبه کد الگوریتم ژنتیک را نشان می‌دهد.

Given:
-nP: base population size.
-nI: number of iterations.
-rC: rate of crossover.
-rM: rate of mutation.
Generate initial population of size nP.
Evaluate initial population according to the fitness function.
While $(current\_iteration \leq nI)$
//Breed $rC \times nP$ new solutions.
Select two parent solutions from current population.
Form offspring's solutions via crossover.
IF$(rand(0.0, 1.0) < rM)$
Mutate the offspring's solutions.
end IF
Evaluate each child solution according to the fitness function.
Add offspring's to population.
//population size is now MaxPop=nP× (1+rC).
Remove the rC× nP least-fit solutions from population.
end While
Output the global best solution

شکل ۳. شبه کد الگوریتم ژنتیک [۴۰].

۳-۲-۲. الگوریتم بهینه‌سازی گرگ خاکستری

بهینه‌سازی گرگ خاکستری یک الگوریتم فرااکتشافی است که رفتار اجتماعی و سلسله‌مراتب گرگ‌های خاکستری را هنگام شکار تقلید می‌کند و اولین بار توسط میرجلیلی و همکاران در سال ۲۰۱۴ ارائه شد. این الگوریتم قادر به حل مسائل بزرگ مقیاس است. روند شکار گرگ‌های خاکستری سه مرحله دارد: ردیابی و نزدیک‌شدن[۵]، تعقیب و محاصره[۶] و حمله[۷] [۴۱]. در سال‌های اخیر، روش‌های مختلفی بر اساس این الگوریتم برای حل مسئله انتخاب ویژگی ارائه شده است، مانند bGWO [۴۲]، GWO-ANN [۴۳].

GWO یک الگوریتم بهینه‌سازی نسبتاً جدیدتر در مقایسه با GA است که به سال ۱۹۷۵ باز می‌گردد. از زمان معرفی، GWO به طور گسترده مورد استفاده قرار گرفته و همچنین برای استفاده در زمینه‌های مختلف مطالعاتی، تغییراتی اعمال شده است. به طور کلی، GWO را می‌توان به چهار نسخه مختلف طبقه‌بندی کرد: ترکیبی[۸]، موازی[۹]، چند هدفه[۱۰] و اصلاح‌شده[۱۱] [۴۴].

در دسته‌ای از گرگ‌های خاکستری، رتبه رهبری به ترتیب نزولی عبارت است از [۴۴, ۴۵]:

۱) گرگ‌های خاکستری آلفا (α)

۲) گرگ‌های خاکستری بتا (β)

۳) گرگ‌های خاکستری دلتا (δ)

۴) گرگ‌های خاکستری امگا (ω)

شبه کد GWO، در شکل (۴) نشان‌داده‌شده است. در این شکل، معادله ۱۲ همان معادله (۳) و معادلات ۴ و ۵ به ترتیب معادلات (۴) و (۵) هستند.

$$\vec{X}(t+1) = \frac{\vec{X_1}+\vec{X_2}+\vec{X_3}}{3} \quad (3)$$

$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a} \quad (4)$$

$$\vec{C} = 2.\vec{r_2} \quad (5)$$

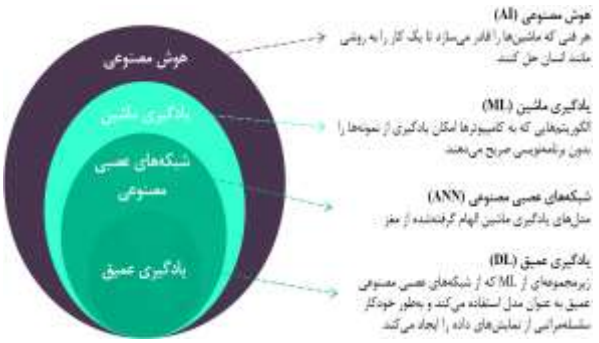
1: Randomly initialize the population of the grey wolves X_i (i=1,2...n)
2: Initialize a, A and C
3: Calculate the fitness value of every grey wolf
4: Rank the grey wolves according to their fitness and name the best, second best and third best grey wolves as X_α, X_β and X_δ respectively
5: **while** termination condition not satisfied **do**
6: Update each grey wolf position using Equation (12)
7: Update value of a, A and C using Equations (4), (5) respectively
8: Evaluate fitness of each grey wolf and update X_α, X_β and X_δ.
9: **end while**
10: Return best solution

شکل ۴. شبه کد الگوریتم بهینه‌سازی گرگ خاکستری [۴۴]

۳-۳. شبکه عصبی مصنوعی

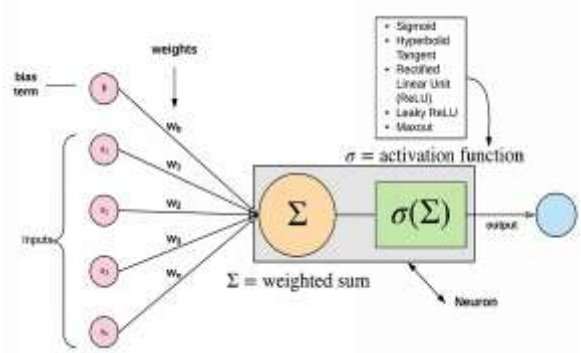
شبکه‌های عصبی مصنوعی[۱۲] یکی از روش‌ها و الگوریتم‌های یادگیری ماشین است که به‌منظور طبقه‌بندی از آن استفاده می‌شود. در شکل (۵)، ارتباط چهار حوزه هوش مصنوعی، یادگیری ماشین، شبکه‌های عصبی مصنوعی و یادگیری عمیق نشان داده شده است.

۱. Meta-Heuristic
۲. Selection
۳. Crossover
۴. Mutation
۵ Tracking and Approaching
۶ Pursuing and Encircling
۷ Attacking
۸. Hybridized
۹. Parallel
۱۰. Multi-objective
۱۱. Modified
۱۲. Artificial neural networks (ANNs)



شکل ۵. ارتباط بین چهار حوزه رایج[۱]

عناصر پردازش شبکه‌های عصبی مصنوعی، نورون‌های مصنوعی[۱] هستند. این نورون‌ها از چهار جزء اساسی تشکیل شده‌اند که شامل داده‌های ورودی، قدرت اتصال (وزن) [۲]، یک تابع فعال‌ساز[۳] (تابع انتقال) و مقادیر خروجی است. ورودی‌های هر نورون در عناصر وزن تنظیم‌شده ضرب می‌شوند (w_{1j} ... w_{nj}) و این ورودی‌های اصلاح‌شده با مقداری به نام «بایاس» (انحراف) جمع می‌شوند. سپس، خروجی وزن‌دار و جمع‌شده از طریق یک تابع غیرخطی (یک تابع فعال‌ساز) عبور می‌کند تا نتایج تولید شوند [۴۶]. همان‌طور که در شکل (۶) نشان داده شده است، نورون مصنوعی ورودی‌های زیادی دارد اما تنها یک خروجی دارد.



شکل ۶. نحوه کار یک نورون مصنوعی

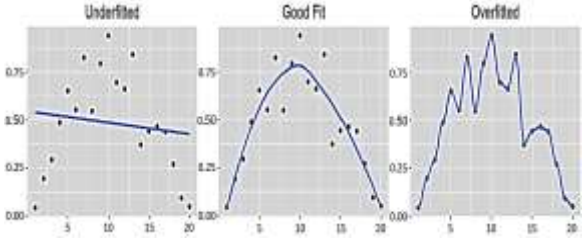
معمولاً، نورون‌ها به‌صورت متصل‌به‌هم در یک ساختار شبکه‌ای چندلایه قرار می‌گیرند که از سه نوع لایه تشکیل شده است: (۱) یک لایه ورودی، (۲) یک یا چند لایه پنهان و (۳) یک لایه خروجی. لایه(های) پنهان، ضرایبی هستند که رابطه بین لایه‌های ورودی و خروجی را فراهم می‌کنند. بر اساس اتصالات بین نورون‌ها و لایه‌ها، شبکه‌های عصبی مصنوعی را می‌توان به دو نوع اصلی تقسیم کرد: شبکه‌های پیشرو[۴] و شبکه‌های بازگشتی[۵] [۴۶].

نقش بایاس در شبکه عصبی این است که خروجی نهایی تابع فعال‌ساز را تغییر دهد. نقش بایاس مشابه نقش مقدار ثابت در تابع خطی است و مقدار نهایی تابع فعال‌ساز را در فضای برداری به چپ و راست منتقل می‌کند تا تابع نهایی بر روی داده‌ها بهتر منطبق شود و مدل در نهایت پیش‌بینی دقیق‌تری داشته باشد.

معمولاً، نورون‌ها به‌صورت متصل‌به‌هم در یک ساختار شبکه‌ای چندلایه قرار می‌گیرند که از سه نوع لایه تشکیل شده است: (۱) یک لایه ورودی، (۲) یک یا چند لایه پنهان و (۳) یک لایه خروجی. لایه(های) پنهان، ضرایبی هستند که رابطه بین لایه‌های ورودی و خروجی را فراهم می‌کنند. بر اساس اتصالات بین نورون‌ها و لایه‌ها، شبکه‌های عصبی مصنوعی را می‌توان به دو نوع اصلی تقسیم کرد: شبکه‌های پیشرو و شبکه‌های بازگشتی [۴۶].

۳-۴. بیش‌برازش و کم‌برازش

بیش برازش و کم برازش[۶] دو چالش مهم در یادگیری ماشین هستند. کم برازش زمانی اتفاق می‌افتد که مدل به‌خوبی با داده‌ها مطابقت نداشته باشد و در نتیجه قادر به ثبت روند در داده‌ها نباشد. بیش برازش زمانی اتفاق می‌افتد که مدل به‌جای درک روند موجود در داده‌ها، نویزها را توصیف کند [۴۷]. بیش‌برازش و کم‌برازش در شکل (۷) نشان داده شده است.



شکل ۷. تفاوت بیش‌برازش و کم‌برازش [۴۸]

(سمت چپ) خط روند، الگوهای کافی را از داده‌ها یاد نگرفته و نتوانسته روند غالب را به تصویر بکشد (کم برازش). (وسط) خط روند مناسبی برای این مجموعه‌داده است. (راست) خط روند، الگوهای زیادی را یاد گرفته و روند غالب را ازدست‌داده است. این الگوریتم قابل‌تعمیم به داده‌های جدید نیست (بیش برازش) [۴۸].

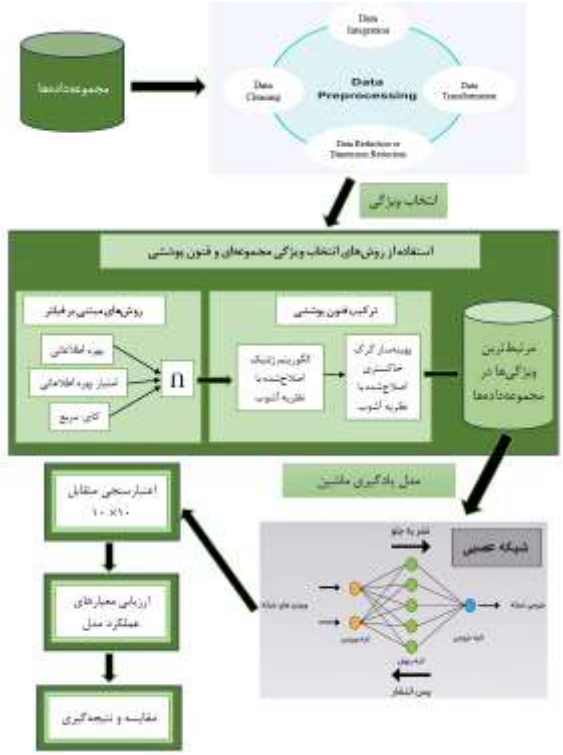
۴. روش تحقیق و مدل پیشنهادی

در این مقاله سعی شده است با بررسی مقالات مرتبط، یک روش انتخاب ویژگی جدید برای افزایش دقت پیش‌بینی کدهای مشکوک نرم‌افزار ارائه شود. مراحل انجام تحقیق حاضر به‌صورت زیر است:

- بررسی و مطالعهٔ جدیدترین مقالات پیش‌بینی کدهای مشکوک نرم‌افزار
- بررسی راه‌های بهبود عملکرد مدل پیش‌بینی کدهای مشکوک در مقالات بررسی‌شده
- تأثیر انتخاب ویژگی در عملکرد پیش‌بینی کدهای مشکوک در مقالات بررسی‌شده
- پیاده‌سازی و اجرای الگوریتم انتخاب ویژگی پیشنهادی روی مجموعه‌داده
- ارزیابی نتایج روش پیشنهادی و مقایسهٔ آن با دیگر روش‌های پیاده‌سازی شده

۱. Artificial Neurons
۲. Connection Strengths (Weights)
۳. Activation Function (Transfer Function)
۴. Feed-Forward Networks
۵. Recurrent Networks
۶. Underfitting

فرآیند کلی روش پیشنهادی در شکل (۸) نشان‌داده‌شده است. در این رویکرد و روش پیشنهادی، یک چارچوب پیش‌بینی کد مشکوک در زبان برنامه‌نویسی پایتون، بر اساس الگوریتم‌های انتخاب ویژگی و فنون یادگیری ماشین ارائه می‌شود.



شکل ۸. فرآیند کلی روش پیشنهادی

۴-۱. مجموعه‌داده

در این مقاله، برای ۵ کد مشکوک متد طولانی، لیست پارامترهای طولانی، کلاس بزرگ، لیست طولانی از کلاس‌های پایه و زنجیره دامنه طولانی، ۵ دیتاست جداگانه استفاده شده است که اولین بار در سال ۲۰۱۸ توسط چن و همکاران تولید شده و در سال ۲۰۲۲ توسط واتاناپاکورن و همکاران بهبود داده شده است.

واتاناپاکورن و همکاران ابتدا همه پروژه‌های منبع‌باز مطابق با نسخه‌های پروژه در کار آقای چن و همکاران را جمع‌آوری کردند. سپس برای بهبود این مجموعه‌داده، ۶۱ معیار مختلف نرم‌افزار را در سطح کلاس و سطح متد از ۱۱۵ پروژه منبع‌باز با استفاده از ابزارهای تحلیل کد استاتیک استخراج کردند؛ یعنی Pysmell، Understand (SciTools) و Cohesion.

علاوه بر این، همچنین برنامه دست‌ساز خود را برای استخراج دو نوع دیگر از معیارهای نرم‌افزاری از کد منبع توسعه دادند: NP (تعداد پارامترهای یک تابع[۱] شامل آرگومان‌های پیش‌فرض، آرگومان‌های غیر کلیدواژه‌ای و آرگومان‌های کلیدواژه‌ای) و NPW (تعداد پارامترهای یک تابع بدون[۲] آرگومان‌های غیر کلیدواژه‌ای و آرگومان‌های کلیدواژه‌ای).

فهرستی از ویژگی‌های نرم‌افزاری استخراج‌شده از تمام ابزارهای تحلیل کد استاتیک در جدول (۲) ارائه شده است.

جدول ۲- ویژگی‌های نرم‌افزاری استخراج‌شده [۱۴]

ابزار	سطح کلاس	سطح متد
Pysmell	Class line of code (CLOC) Number of base classes (NBC)	Method line of code (MLOC) Number of parameters (PAR) Depth of closure (DOC)
Understand	AvgCyclomatic AvgCyclomaticModified AvgCyclomaticStrict AvgEssential AvgLine AvgLineBlank AvgLineCode (AMS) AvgLineComment CountClassBase (IFANIN) CountClassCoupled (CBO) CountClassCoupledModified CountClassDerived (NOC) CountDeclInstanceMethod (NIM) CountDeclInstanceVariable (NIV) CountDeclMethod (WMC) CountDeclMethodAll (NM) CountLine (NL) CountLineBlank (BLOC) CountLineCode (LOC, SLOC) CountLineCodeDecl CountLineCodeExe CountLineComment CountStmt CountStmtDecl CountStmtExe MaxCyclomatic MaxCyclomaticModified MaxCyclomaticStrict MaxEssential MaxInheritanceTree (DIT) MaxNesting RatioCommentToCode SumCyclomatic (WCM) SumCyclomaticModified SumCyclomaticStrict SumEssential	CountLine (NL) CountLineBlank (BLOC) CountLineCode (LOC, SLOC) CountLineCodeDecl CountLineCodeExe CountLineComment CountPath (NPATH) CountPathLog CountStmt CountStmtDecl CountStmtExe Cyclomatic (V(G)) CyclomaticModified CyclomaticStrict Essential (EV(G)) MaxNesting RatioCommentToCode
Cohesion	Cohesion	-
برنامه دست‌ساز واتاناپاکورن و همکاران	-	Number of parameter including default and arbitrary(*,**) arguments (NP) Number of parameter without arbitrary(*,**) arguments (NPW)

۴-۲. پیش‌پردازش

در مرحله دوم، عملیات پیش‌پردازش بر روی مجموعه‌داده انجام می‌گیرد. در این بخش عملیاتی نظیر نرمال‌سازی، پاک‌سازی و

۱. Number of Parameters for a Function
۲. Number of Parameters for a Function Without

متعادل‌سازی انجام می‌شود. هدف از این بخش بهبود داده‌ها قبل از عملیات انتخاب ویژگی است.

در ابتدای مرحله پیش‌پردازش، داده‌های موجود در مجموعه‌داده باید نرمال‌سازی شوند. با اینکه به طور عمومی نیاز نیست همه مجموعه‌داده‌ها مورد نرمال‌سازی قرار بگیرند، ولی چون محدوده بعضی از ویژگی‌ها در مجموعه‌داده با محدوده برخی دیگر از ویژگی‌های موجود در مجموعه‌داده متفاوت است، نیاز به نرمال‌سازی وجود دارد تا مدل یادگیری ماشین بتواند آن‌ها را مورد پردازش قرار دهد. برای این منظور می‌توان از فنون مختلف نرمال‌سازی نظیر min-max، استفاده کرد.

در مرحله دوم؛ چون مجموعه‌داده‌ها معمولاً شامل مقادیر ازدست‌رفته است، نیاز به مدیریت این مقادیر است. در صورت نیاز برای برطرف‌سازی این مشکل می‌توان از روش‌های مختلف مدیریت داده‌های ازدست‌رفته نظیر حذف[۱] سطرها یا ستون‌هایی با مقادیر ازدست‌رفته یا جایگزینی با مقادیر میانگین/میانه/نما (مد) [۲] استفاده کرد.

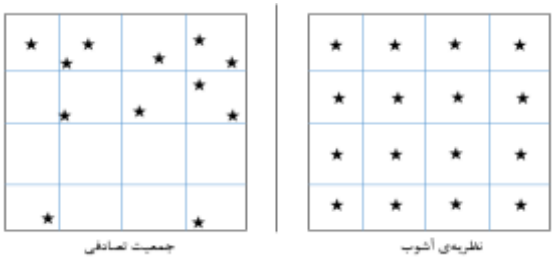
مرحله بعدی پیش‌پردازش، متعادل‌سازی مجموعه‌داده‌هاست که در مجموعه‌داده‌هایی که تعداد نمونه‌های مثبت و منفی متعادل نباشد، در صورت نیاز می‌توان از الگوریتم‌هایی مانند SMOTE استفاده کرد. در این مقاله از روش SMOTE با اعمال محدودیت ۳۳٪ استفاده شده است. یعنی بعد از اعمال این روش، تعداد نمونه‌های اقلیت که نمونه‌های مثبت هستند، برابر با ۳۳٪ تعداد نمونه‌های اکثریت شود.

۴-۳. انتخاب ویژگی

در گام سوم، برای افزایش دقت مدل‌های پیش‌بینی خطای نرم‌افزار، باید ویژگی‌های مرتبط از مجموعه‌داده‌ها را انتخاب نمود. برای انتخاب ویژگی‌های مرتبط روش‌های متعددی مطرح شده است که ازجمله آنها می‌توان به روش‌های فیلتر و پوشش اشاره کرد. روش‌های فیلتر روش‌های آماری بوده که مهم‌ترین خصیصه این روش‌ها بدون ناظر بودن آنها است. مهم‌ترین ضعف این روش‌ها درنظرنگرفتن وابستگی بین ویژگی‌ها است. به عبارتی در این روش‌ها، اگر وابستگی بین داده‌ها وجود داشته باشد، کارایی روش پایین می‌آید. روش‌های پوشش جزء روش‌های با ناظر محسوب می‌شوند که بر اساس قوانین طبیعت به دنبال یافتن زیرمجموعه‌ای مناسب از ویژگی‌ها هستند و مهم‌ترین ضعف آنها زمان‌بر بودن و پیچیدگی نسبت به الگوریتم‌های مبتنی بر فیلتر است؛ لذا در این مقاله، با بهره‌گیری از مزایای دو روش فیلتر و پوشش و ارائه روشی جدید، به دنبال استخراج ویژگی‌هایی هستیم که علاوه بر داشتن بالاترین رتبه[۳] بین ویژگی‌های مجموعه‌داده (انتخاب توسط الگوریتم‌های مبتنی بر فیلتر) و یافتن بهترین ترکیب ممکن از ویژگی‌های منتخب در مرحله قبل توسط فنون مبتنی بر پوشش، باعث افزایش کارایی سیستم پیش‌بینی نیز گردد.

لذا برای این منظور، ابتدا از فن انتخاب ویژگی مجموعه‌ای که شامل الگوریتم‌های انتخاب ویژگی مبتنی بر فیلتر بهره اطلاعاتی، امتیاز بهره اطلاعاتی و کای-مربع است، استفاده کرده و پس از اشتراک‌گیری از نتایج سه الگوریتم برای به‌دست‌آوردن بهینه‌ترین ویژگی‌ها، آن را به‌عنوان ورودی به الگوریتم ژنتیک می‌دهیم. در این مرحله برای تولید جمعیت اولیه در الگوریتم ژنتیک، از نظریه آشوب استفاده خواهیم کرد که مهم‌ترین مزیت این کار، پوشش‌دهی بیشتر فضای مسئله و ایجاد تنوع در جمعیت اولیه است. لازم به ذکر است که جمعیت اولیه، نقش کلیدی و اساسی در موفقیت یک الگوریتم دارد. در شکل (۹)، تفاوت نظریه آشوب و جمعیت تصادفی نشان داده شده است.

الگوریتم GA اغلب با سایر الگوریتم‌های بهینه‌سازی ترکیب می‌شود. مزایایی که از ترکیب‌شدن GA به دست می‌آید، بهره‌وری[۴] بهتر، روش‌های باکیفیت بهتر و پارامترهای کنترلی بهینه است. سپس خروجی الگوریتم ژنتیک را به‌عنوان ورودی بهینه‌ساز گرگ خاکستری اصلاح‌شده با نظریه آشوب استفاده می‌کنیم. مزیت ترکیب دو الگوریتم GA و GWO عبارت‌اند از: GA اصلی با که با نظریه آشوب اصلاح شده است، به بهبود اکتشاف فضای جستجو کمک می‌کند درحالی‌که استفاده از GWO اصلاح‌شده با نظریه آشوب، بهره‌برداری مناسب از فضای جستجو را تضمین می‌کند. اعمال فنون انتخاب ویژگی باعث بهبود عملکرد و کاهش زمان آموزش مدل یادگیری ماشین می‌شود.



شکل ۹. تفاوت نظریه آشوب و جمعیت تصادفی

بهبود اکتشاف فضای جست‌وجو و بهره‌برداری مناسب از آن در مسائل بهینه‌سازی به موضوعات مهمی اشاره دارند. اکتشاف به توانایی الگوریتم در بررسی و یافتن نقاط جدید در فضای جست‌وجو اشاره دارد. هدف از اکتشاف، یافتن نقاط امیدبخش و نواحی بهینه در فضای جست‌وجو است که ممکن است در ابتدا ناشناخته باشند. اکتشاف مناسب به الگوریتم کمک می‌کند تا از افتادن در بهینه‌های محلی جلوگیری کند و به بهینه‌های سراسری نزدیک شود. بهره‌برداری به توانایی الگوریتم در استفاده از اطلاعات موجود برای بهبود جواب‌های کنونی اشاره دارد. هدف از

۱. Deletion
۲. Mean/Median/Mode Imputation
۳. Highest-Ranked
٤. Efficiency

بهره‌برداری، بهبود و تصفیه جواب‌های موجود برای رسیدن به نتیجه بهینه‌تر است. بهره‌برداری مناسب به الگوریتم کمک می‌کند تا بتواند جواب‌های موجود را به طور مؤثر بهبود بخشد.

مقادیر فراپارامترها در الگوریتم ژنتیک عبارت‌اند از:

اندازه جمعیت[۱] = ۱۰۰، تعداد نسل‌ها[۲] = ۲۰، نرخ ترکیب = ۰/۷ و نرخ جهش = ۰/۲.

مقادیر فراپارامترها در الگوریتم گرگ خاکستری عبارت‌اند از:

تعداد گرگ‌ها (عامل‌ها) [۳] = ۳۰، حداکثر تکرار[۴] = ۱۵، مقدار حد پایین[۵] = ۰ و مقدار حد بالا[۶] = ۱

۴-۴. مدل طبقه‌بندی

در این مرحله، پس از به‌دست‌آوردن بهترین ویژگی‌ها، نوبت به طراحی مدل طبقه‌بند می‌رسد. در این مقاله از شبکه عصبی پرسپترون چندلایه[۷] پس انتشار خطا به‌عنوان طبقه‌بند استفاده خواهد شد.

در این شبکه عصبی از تابع فعال‌ساز ReLU و الگوریتم بهینه‌ساز Adam استفاده شده است و حداکثر تعداد تکرار جهت بهینه‌سازی و به‌روزرسانی وزن‌های شبکه برابر ۲۰۰۰ در نظر گرفته شده است. همچنین از روش جستجوی شبکه‌ای برای تعیین بهترین تعداد لایه در بازه ۳ تا ۱۱ لایه استفاده می‌شود. به‌علاوه، تعداد نورون‌های لایه‌های مخفی باید با تعداد ویژگی‌های لایه ورودی برابر باشد. اگر تعداد نورون‌های لایه مخفی کمتری انتخاب شود، منجر به کم برازش می‌شود. در حالی که اگر تعداد زیادی نورون را انتخاب کنیم ممکن است منجر به بیش‌برازش، واریانس بالا و افزایش زمان لازم برای آموزش شبکه شود. تابع ReLU یک تابع فعال‌ساز غیرخطی است که به‌صورت زیر تعریف می‌شود:

$$f(x) = \max(0, x)$$

به عبارت دیگر، ReLU خروجی صفر را برای مقادیر ورودی منفی و خروجی برابر با ورودی را برای مقادیر ورودی مثبت ارائه می‌دهد. ReLU دارای چندین مزیت است:

- ✓ محاسبات ساده‌تر و سریع‌تر نسبت به توابع غیرخطی دیگر مانند Sigmoid و Tanh.
- ✓ جلوگیری از مشکل محوشدگی گرادیان[۸] که در تابع Sigmoidرخ می‌دهد.
- ✓ ایجاد پراکندگی[۹] در لایه‌های پنهان شبکه عصبی که به بهبود عملکرد مدل کمک می‌کند.

الگوریتم Adam یک الگوریتم بهینه‌سازی مبتنی بر گرادیان است که برای مسائل بهینه‌سازی غیرخطی و بدون هرگونه محدودیت قیدی (بدون محدودیت در متغیرهای تصمیم‌گیری مثل محدودیت‌های مساوی یا نامساوی) به کار می‌رود. این الگوریتم با استفاده از گرادیان‌های لحظه‌ای و میانگین متحرک آنها، سرعت همگرایی را افزایش می‌دهد. بهینه‌ساز Adam ترکیبی از مفاهیم دو بهینه‌ساز RMSProp و Momentum است:

- ❖ RMSProp برای تنظیم نرخ یادگیری هر پارامتر به‌صورت جداگانه استفاده می‌شود.
- ❖ Momentum برای کاهش نوسانات گرادیان و تسریع همگرایی به کار می‌رود.

بهینه‌ساز Adam به طور پویا نرخ یادگیری را برای هر پارامتر به‌روز می‌کند و به همین دلیل در مقایسه با الگوریتم‌های دیگر مانند SGD، سریع‌تر همگرا می‌شود. در مجموع، ReLU و Adam دو انتخاب رایج و موفق هستند که به طور گسترده در شبکه‌های عصبی استفاده می‌شوند.

لازم به ذکر است که مدل طبقه‌بند شبکه عصبی با ویژگی‌های گفته‌شده و دارای ۵ لایه مخفی، در مراحل انتخاب ویژگی توسط الگوریتم‌های ژنتیک و گرگ خاکستری نیز به‌عنوان تابع برازش[۱۰] مورد استفاده قرار گرفته و بهترین ترکیب از ویژگی‌ها را به‌عنوان خروجی نهایی هر مرحله بر اساس معیار دقت ارائه می‌دهد. آموزش و آزمون تابع برازش نیز همانند طبقه‌بند نهایی که در بخش بعدی ارائه می‌شود، انجام شده است.

۴-۵. آموزش و آزمون طبقه‌بند

سپس نوبت به آموزش و آزمون مدل شبکه عصبی برای شناسایی کدهای مشکوک می‌رسد. برای آموزش و آزمون شبکه عصبی، از روش «اعتبارسنجی متقابل k-بخشی طبقه‌بندی‌شده[۱۱]» که یکی از انواع اعتبارسنجی متقابل است، استفاده می‌شود. بدین منظور، با تقسیم داده‌ها به ۱۰ قسمت، ۱۰ بار تکرار خواهیم داشت. در هر تکرار، یک قسمت متفاوت از قسمت‌های قبلی داده‌ها به‌عنوان مجموعه‌داده آزمون در نظر گرفته می‌شود و قسمت‌های دیگر برای داده‌های آموزش است. اعتبارسنجی متقابل یک فن اعتبارسنجی مدل رایج در تشخیص کد مشکوک است.

همان‌طور که گفته شد، در این مقاله از روش اعتبارسنجی متقابل ۱۰-بخشی طبقه‌بندی‌شده، بهره گرفته می‌شود. KFold یک اعتبارسنجی متقابل است که مجموعه‌داده را به k-بخش تقسیم می‌کند. اعتبارسنجی متقابل k-بخشی طبقه‌بندی‌شده یعنی اطمینان حاصل شود که هر بخش از مجموعه‌داده دارای

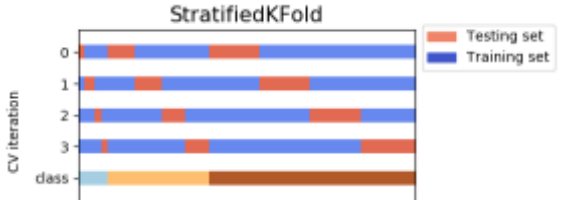
۱. Population Size
۲. Number of Generations
۳. Number of Wolves
۴. Maximum Iteration
۵. Lower Bound (Lb)
۶. Upper Bound (Ub)
۷. Multilayer Perceptron (MLP)
۸. Vanishing Gradient
۹. Sparsity
۱۰. Fitness Function
۱۱. Stratified K-Fold Cross-Validation

نسبت مشاهدات یکسانی با یک برچسب مشخص است؛ بنابراین، به این معنی است که KFold طبقه‌بندی‌شده، نسخه بهبودیافته KFold است.

لذا، هنگام مواجهه با وظایف طبقه‌بندی با توزیع نامتوازن کلاس‌ها (مجموعه‌داده‌های نامتعادل)، باید StratifiedKFold را به KFold ترجیح داد. نحوه تقسیم‌بندی داده‌ها در روش k-بخشی طبقه‌بندی‌شده در شکل (۱۰) نشان‌داده‌شده است. به‌عنوان‌مثال، n_splits = ۴ و داده‌ها دارای ۳ کلاس (برچسب) برای y (متغیر وابسته) است. ۴ مجموعه‌آزمون، تمام داده‌ها را بدون هیچ هم‌پوشانی پوشش می‌دهند.

۴-۶. اعتبارسنجی مدل طبقه‌بند

در گام ششم، باید مدل آموزش‌داده‌شده، مورد ارزیابی قرار گیرد. به دلیل اینکه در این مقاله از روش اعتبارسنجی متقابل ۱۰×۱۰ طبقه‌بندی‌شده استفاده می‌شود، باید در هر مرحله پس از آموزش مدل، عملکرد مدل را با استفاده از داده‌های آزمون، ارزیابی و معیارهای عملکرد مدل را محاسبه کرد. پس از ۱۰ بار تکرار، می‌توان معیارهای عملکرد مدل را با میانگین‌گیری از معیارها در هر بار ارزیابی محاسبه کرد. این معیارها شامل دقت، صحت، پوشش، معیار F و ضریب همبستگی متیوز (MCC) است.



شکل ۱۰. مثالی از نحوه عملکرد اعتبارسنجی متقابل k-بخشی طبقه‌بندی‌شده (۴-بخش)[۱]

۴-۷. دریافت خروجی مراحل روش پیشنهادی

مرحله هفتم که گام نهایی تحقیق است، مرحله استفاده از مجموعه‌داده‌ها، پیاده‌سازی عملی هر یک از مراحل مذکور بر روی مجموعه‌داده‌ها و آموزش و اعتبارسنجی مدل توسط بهترین زیرمجموعه از ویژگی‌ها است. در نهایت می‌بایست نتایج حاصل که به کمک روش اعتبارسنجی متقابل ۱۰-بخشی طبقه‌بندی‌شده به‌دست‌آمده، مورد بررسی قرار گیرد. مطالب مربوط به خروجی نهایی مدل و مقایسه مقادیر ۴ معیار دقت، صحت، پوشش و معیار F با مقاله مرجع در فصل پنجم ارائه خواهد شد. علاوه بر ۴ معیار گفته‌شده، معیار دیگری نیز به نام ضریب همبستگی متیوز استفاده شده است که یک معیار ایده‌آل برای مجموعه‌داده‌های نامتعادل در نظر گرفته می‌شود.

خروجی مراحل در چهار حالت برای هر مجموعه‌داده مربوط به کد مشکوک (متد طولانی، لیست پارامترهای طولانی، زنجیره دامنه طولانی، کلاس بزرگ و لیست طولانی از کلاس‌های پایه) ارائه خواهد شد:

۱. بدون هرگونه تغییر در مجموعه‌داده و استفاده از آن برای آموزش و اعتبارسنجی مدل طبقه‌بند

۲. بدون پیش‌پردازش، انجام مراحل انتخاب ویژگی و استفاده از بهترین ویژگی‌ها برای آموزش و اعتبارسنجی مدل طبقه‌بند

۳. پیش‌پردازش بدون استفاده از روش SMOTE، انجام مراحل انتخاب ویژگی و استفاده از بهترین ویژگی‌ها برای آموزش و اعتبارسنجی مدل طبقه‌بند

۴. پیش‌پردازش با استفاده از روش SMOTE، انجام مراحل انتخاب ویژگی و استفاده از بهترین ویژگی‌ها برای آموزش و اعتبارسنجی مدل طبقه‌بند

۵. نتایج و بحث

در فاز اول این تحقیق برای بهبود عملکرد مدل پیش‌بینی کدهای مشکوک، از روش‌های انتخاب ویژگی مجموعه‌ای و فنون پوششی استفاده می‌شود که جزئیات آن در بخش ۴-۳ ارائه شده است.

در فاز دوم، بعد از به‌دست‌آوردن خروجی نهایی مرحله انتخاب ویژگی که شامل بهترین ویژگی‌ها است، فرآیند آموزش و اعتبارسنجی به کمک روش اعتبارسنجی متقابل ۱۰-بخشی طبقه‌بندی‌شده آغاز می‌شود. در این فاز از شبکه عصبی ساخته‌شده که پرسپترون چندلایه پس انتشار خطا است، به‌عنوان مدل طبقه‌بند استفاده می‌شود. چهار معیار اصلی دقت، صحت، پوشش و معیار F به همراه معیار MCC جهت ارزیابی مدل مورد توجه قرار گرفته‌اند. جزئیات مربوط به شبکه عصبی و فرآیند آموزش و اعتبارسنجی در بخش‌های ۴-۴، ۴-۵ و ۴-۶ تشریح گردیده است.

در این بخش، ابتدا مراحل انجام تحقیق و نتایج ارزیابی روش واتاناپاکورن و همکاران ارائه شده و سپس نتایج روش پیشنهادی در این مقاله، به‌صورت کامل و در یک جدول بیان می‌شود. در نهایت مقایسه دو روش، ارائه خواهد شد.

تحقیق واتانپاکورن و همکاران شامل سه بخش اصلی است:

۱. یک مجموعه‌داده را با جمع‌آوری ۶۱ معیار نرم‌افزار در سطح کلاس و متد از ۱۱۵ پروژه منبع باز پایتون با استفاده از ابزارهای تحلیل کد ایستا ایجاد کردند که در بخش ۳-۱ ارائه شده است.

۲. از فنون انتخاب ویژگی، مانند انتخاب ویژگی مبتنی بر همبستگی[۲] و رگرسیون لجستیک با روش گام‌به‌گام روبه‌جلو[۳] برای یافتن مرتبط‌ترین ویژگی‌ها برای هر نوع کد مشکوک استفاده کردند. جدول (۳) نتایج به‌دست‌آمده از فنون انتخاب ویژگی را نشان می‌دهد.

۱. https://stackoverflow.com/questions/٤٥٩٦٩٣٩٠/difference-between-stratifiedkfold-and-stratifiedshufflesplit-in-sklearn

۲. Correlation-based Feature Selection (CFS)
۳. Logistic Regression-Forward Stepwise (LRFS)

۳. آنها از هشت مدل یادگیری ماشین برای تشخیص پنج نوع کد مشکوک استفاده کردند. همچنین، نتایج را با نتایج حاصل از روش تنظیم ماشین در کار چن و همکاران مقایسه کردند.

هشت مدل یادگیری ماشین نظارت شده عبارت‌اند از: درخت تصمیم، درختان گرادیان تقویت‌شده، جنگل تصادفی، ماشین بردار پشتیبان، k-نزدیک‌ترین همسایه، رگرسیون لجستیک، پرسپترون چندلایه و بیز ساده. همچنین از کتابخانه scikit-learn پایتون برای پیاده‌سازی آنها استفاده کردند.

چن و همکاران روش ماشین تنظیم را پیشنهاد کردند که رویکردی برای یافتن خودکار مقادیر آستانه مناسب و تنظیم راهبردهای تشخیص است. این بر اساس یک مخزن نمونه است که قطعات طراحی را جمع‌آوری می‌کند که به‌عنوان کد مشکوک پایتون شناسایی شده‌اند یا خیر. سپس مخزن به ماشین تنظیم اجازه می‌دهد تا مقادیر آستانه‌ای را انتخاب کند که صحت و پوشش نمونه‌های شناسایی‌شده را به حداکثر می‌رساند. این کار آسان نیست؛ زیرا به نمونه‌های صحیح به‌اندازه کافی بزرگ نیاز دارد که به‌صورت دستی شناسایی شده‌اند.

آنها در آزمایش‌های خود از دو گروه پیکربندی استفاده کردند. در گروه اول، از مقادیر پارامترهای پیش‌فرض scikit-learn استفاده نمودند. جدول (۴) گروه دیگری از پیکربندی‌ها را ارائه می‌کند که در آن مجموعه‌ای از فراپارامترها را برای هر مدل یادگیری ماشین مشخص کرده و از روش جستجوی شبکه‌ای با اعتبارسنجی متقابل تودرتو بر اساس مجموعه‌ای از فراپارامترها برای یافتن بهترین مقادیر فراپارامتر برای مدل‌های یادگیری ماشین و همچنین بهبود عملکرد استفاده کردند.

جدول‌های (۵) و (۶) نتایج ارزیابی عملکرد طبقه‌بندی‌کننده‌های یادگیری ماشین را نشان می‌دهند. در جدول‌های (۵) و (۶)، "*" نشان می‌دهد که مقادیر پارامتر پیش فرض scikit-learn استفاده شده است. "**" نشان می‌دهد که فراپارامترهای جدول (۴) برای مدل مربوطه اعمال شده است.

ویژگی‌های انتخاب‌شده در پایان مراحل انتخاب ویژگی برای هر کد مشکوک در جدول (۷) ارائه شده است. نتایج به‌دست‌آمده از هر مجموعه‌داده مربوط به ۵ کد مشکوک، بدون اعمال مراحل پیش‌پردازش و انتخاب ویژگی و با استفاده از پرسپترون چندلایه پس انتشار خطا که در این مقاله استفاده شده است، در جدول (۸) ارائه شده است.

جدول (۹) مقایسه نتایج روش واتاناپاکورن و همکاران با کار چن و همکاران و همچنین با روش پیشنهادی در این مقاله (شامل پیش‌پردازش، انتخاب ویژگی و استفاده از شبکه عصبی) ارائه شده است.

جدول ۳- نتایج انتخاب ویژگی در تحقیق واتاناپاکورن و همکاران [۱۴]

کد مشکوک	سطح	چن و همکاران	CFS	LRFS
LM	متد	MLOC	CountStmtExe EV(G) MaxNesting NP MLOC	MLOC
LC	کلاس	CLOC	NL CLOC	CLOC
LPL	متد	PAR	NPW PAR	NP PAR
LSC	متد	DOC	CountStmt CountStmtDecl DOC	CountLineCodeDecl DOC
LBCL	کلاس	NBC	CountLineCodeDecl NBC	NBC

جدول ۴- مجموعه فراپارامترهای مورد استفاده در تحقیق واتاناپاکورن و همکاران [۱۴]

مدل طبقه‌بند	فراپارامترها
درخت تصمیم	'max depth' : [None, ۲, ۴, ۶, ۸, ۱۰, ۱۲] 'criterion' : ['gini', 'entropy']
درختان گرادیان تقویت‌شده	'learning rate': [۰.۱, ۰.۰۱] 'n estimators' : [۱۰۰, ۵۰۰, ۱۰۰۰] 'max depth' : [۳, ۴, ۶, ۸, ۱۰, ۱۲]
جنگل تصادفی	'n estimators': [۵, ۲۰, ۵۰, ۱۰۰] 'max features': ['auto', 'sqrt'] 'max depth': [None, ۱۰, ۲۰, ۳۰, ۴۰, ۵۰, ۶۰, ۷۰, ۸۰, ۹۰, ۱۰۰]
ماشین بردار پشتیبان	'C':list(range(−۱, ۱۱))
k-نزدیک‌ترین همسایه	'n neighbors' : list(range(۱, ۱۰)) 'leaf size' : list(range(۱, ۵۰)) 'p':[۱, ۲]
رگرسیون لجستیک	'C':list(range(−۱ ,۱۱)) 'penalty' : ['l۱', 'l۲'] 'max iter': [۲۰, ۵۰, ۱۰۰, ۲۰۰, ۵۰۰, ۱۰۰۰] 'solver': ['lbfgs', 'liblinear'] 'class weight': [None,'balanced']
پرسپترون چندلایه	'alpha': [۰.۰۰۰۱, ۰.۰۵] 'learning rate': ['constant','adaptive']
بیز ساده	'var smoothing': [۱e − ۱۱, ۱e − ۱۰, ۱e − ۹]

جدول ۵- نتایج ارزیابی عملکرد طبقه‌بندی‌کننده‌های یادگیری ماشین در تحقیق واتاناپاکورن و همکاران (۱) [۱۴]

طبقه‌بند	ویژگی‌ها	LM			LPL			LSC		
		Precis. (%)	Recall (%)	F-measure (%)	Precis. (%)	Recall (%)	F-measure (%)	Precis. (%)	Recall (%)	F-measure (%)
K-NN	All	۹۲/۰۰	۸۵/۱۹	۸۸/۴۶*	۵۶/۷۶	۳۳/۳۳	۴۲/۰۰**	۴۱/۶۷	۲۰/۸۳	۲۷/۷۸**
	Chen et al.	۹۲/۸۶	۹۶/۳۰	۹۴/۵۵**	۹۶/۲۳	۸۰/۹۵	۸۷/۹۳	۱۰۰/۰۰	۷۰/۸۳	۸۲/۹۳
	CFS	۹۶/۳۰	۹۶/۳۰	۹۶/۳۰*	۸۹/۶۶	۸۵/۵۴	۸۵/۹۵*	۶۸/۷۵	۴۵/۸۳	۵۵/۰۰**
	LRFS	۹۲/۸۶	۹۶/۳۰	۹۴/۵۵**	۹۰/۰۰	۸۵/۷۱	۸۷/۸۰*	۸۹/۴۷	۷۰/۸۳	۷۹/۰۷**
LR	All	۸۵/۱۹	۸۵/۱۹	۸۵/۱۹**	۸۸/۸۹	۸۸/۸۹	۸۸/۸۹**	۹۰/۴۸	۷۹/۱۷	۸۴/۴۴**
	Chen et al.	۸۹/۶۶	۹۶/۳۰	۹۲/۸۶**	۹۶/۲۳	۸۰/۹۵	۸۷/۹۳	۱۰۰/۰۰	۷۰/۸۳	۸۲/۹۳
	CFS	۸۹/۶۶	۹۶/۳۰	۹۲/۸۶**	۹۶/۴۹	۸۷/۳۰	۹۱/۶۷*	۹۵/۴۵	۸۷/۵۰	۹۱/۳۰*
	LRFS	۸۹/۶۶	۹۶/۳۰	۹۲/۸۶**	۹۴/۸۳	۸۷/۳۰	۹۰/۹۱*	۸۸/۴۶	۹۵/۸۳	۹۲/۰۰**
DT	All	۸۹/۲۹	۹۲/۵۹	۹۰/۹۱	۹۱/۵۳	۸۵/۷۱	۸۸/۵۲**	۹۱/۶۷	۹۱/۶۷	۹۱/۶۷**
	Chen et al.	۹۶/۴۳	۱۰۰/۰۰	۹۸/۱۸*	۹۶/۲۳	۸۰/۹۵	۸۷/۹۳	۱۰۰/۰۰	۷۰/۸۳	۸۲/۹۳
	CFS	۹۲/۵۹	۹۲/۵۹	۹۲/۵۹*	۹۱/۶۷	۸۷/۳۰	۸۹/۴۳*	۹۱/۶۷	۹۱/۶۷	۹/۶۷**
	LRFS	۹۶/۴۳	۱۰۰/۰۰	۹۸/۱۸*	۹۱/۶۷	۸۷/۳۰	۸۹/۴۳*	۱۰۰/۰۰	۹۵/۸۳	۹۷/۸۷
GBT	All	۹۲/۵۹	۹۲/۵۹	۹۲/۵۹*	۸۷/۵۰	۸۸/۸۹	۸۸/۱۹*	۹۵/۶۵	۹۱/۶۷	۹۳/۶۲**
	Chen et al.	۹۶/۴۳	۱۰۰/۰۰	۹۸/۱۸	۹۶/۲۳	۸۰/۹۵	۸۷/۹۳	۱۰۰/۰۰	۷۰/۸۳	۸۲/۹۳
	CFS	۹۲/۵۹	۹۲/۵۹	۹۲/۵۹	۹۱/۶۷	۸۷/۳۰	۸۹/۴۳	۸۶/۹۶	۸۳/۳۳	۸۵/۱۱*
	LRFS	۹۶/۴۳	۱۰۰/۰۰	۹۸/۱۸	۹۱/۶۷	۸۷/۳۰	۸۹/۴۳*	۱۰۰/۰۰	۹۵/۸۳	۹۷/۸۷
RF	All	۹۶/۱۵	۹۲/۵۹	۹۴/۳۴	۸۸/۳۳	۸۴/۱۳	۸۶/۱۸*	۹۰/۹۱	۸۳/۳۳	۸۶/۹۶**
	Chen et al.	۹۶/۴۳	۱۰۰/۰۰	۹۸/۱۸	۹۶/۲۳	۸۰/۹۵	۸۷/۹۳	۱۰۰/۰۰	۷۰/۸۳	۸۲/۹۳
	CFS	۹۲/۵۹	۹۲/۵۹	۹۲/۵۹	۹۱/۶۷	۸۷/۳۰	۸۹/۴۳	۹۱/۳۰	۸۷/۵۰	۸۹/۳۶
	LRFS	۹۶/۴۳	۱۰۰/۰۰	۹۸/۱۸	۹۲/۰۶	۹۲/۰۶	۹۲/۰۶**	۱۰۰/۰۰	۹۵/۸۳	۹۷/۸۷
NB	All	۹۱/۶۷	۴۰/۷۴	۵۶/۴۱**	۱۷/۰۸	۸۷/۳۰	۲۸/۵۷*	۴۰/۰۰	۱۶/۶۷	۲۳/۵۳**
	Chen et al.	۹۰/۰۰	۱۰۰/۰۰	۹۴/۷۴	۹۱/۰۷	۸۰/۹۵	۸۵/۷۱	۱۰۰/۰۰	۷۰/۸۳	۸۲/۹۳
	CFS	۶۴/۲۹	۱۰۰/۰۰	۷۸/۲۶	۸۱/۶۷	۷۷/۷۸	۷۹/۶۷	۹۱/۶۷	۴۵/۸۳	۶۱/۱۱
	LRFS	۹۰/۰۰	۱۰۰/۰۰	۹۴/۷۴	۸۲/۰۹	۸۷/۳۰	۸۴/۶۲	۹۱/۳۰	۸۷/۵۰	۸۹/۳۶
MLP	All	۶۸/۹۷	۷۴/۰۷	۷۱/۴۳*	۴۷/۸۳	۱۷/۴۶	۲۵/۵۸*	۳۵/۲۹	۲۵/۰۰	۲۹/۲۷**
	Chen et al.	۹۶/۱۵	۹۲/۵۹	۹۴/۳۴*	۹۶/۲۳	۸۰/۹۵	۸۷/۹۳	۱۰۰/۰۰	۴/۱۷	۸/۰۰*
	CFS	۹۱/۳۰	۷۷/۷۸	۸۴/۰۰**	۹۵/۹۲	۷۴/۶۰	۸۳/۹۳**	۶۶/۶۷	۸/۳۳	۱۴/۸۱
	LRFS	۹۶/۱۵	۹۲/۵۹	۹۴/۳۴*	۹۶/۳۶	۸۴/۱۳	۸۹/۸۳	۵۰/۰۰	۴/۱۷	۷/۶۹
SVM	All	۱۰۰/۰۰	۷/۴۱	۱۳/۷۹	۰/۰۰	۰/۰۰	۰/۰۰	۰/۰۰	۰/۰۰	۰/۰۰
	Chen et al.	۹۲/۵۹	۹۲/۵۹	۹۲/۵۹	۹۶/۲۳	۸۰/۹۵	۸۷/۹۳	۱۰۰/۰۰	۷۰/۸۳	۸۲/۹۳
	CFS	۹۶/۳۰	۹۶/۳۰	۹۶/۳۰	۹۱/۵۳	۸۵/۷۱	۸۸/۵۲**	۰/۰۰	۰/۰۰	۰/۰۰
	LRFS	۹۲/۵۹	۹۲/۵۹	۹۲/۵۹	۹۱/۵۳	۸۵/۷۱	۸۸/۵۲	۹۰/۰۰	۷۵/۰۰	۸۱/۸۲**

جدول ۶- نتایج ارزیابی عملکرد طبقه‌بندی‌کننده‌های یادگیری ماشین در تحقیق واتاناپاکورن و همکاران (۲) [۱۴]

		LC			LBCL		
طبقه‌بند	ویژگی‌ها	Precis. (%)	Recall (%)	F-measure (%)	Precis. (%)	Recall (%)	F-measure (%)
K-NN	All	۷۸/۵۷	۵۶/۴۱	۶۵/۶۷*	۲۵/۰۰	۳/۲۳	۵/۷۱*
	Chen et al.	۹۲/۵۰	۹۴/۸۷	۹۳/۶۷**	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	CFS	۷۷/۷۸	۷۱/۷۹	۷۴/۶۷**	۸۷/۵۰	۶۷/۷۴	۷۶/۳۶**
	LRFS	۹۲/۵۰	۹۴/۸۷	۹۳/۶۷**	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
LR	All	۷۲/۲۲	۶۶/۶۷	۶۹/۳۳**	۹۰/۶۳	۹۳/۵۵	۹۲/۰۶**
	Chen et al.	۹۴/۱۲	۸۲/۰۵	۸۷/۶۷*	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	CFS	۸۵/۷۱	۷۶/۹۲	۸۱/۰۸*	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	LRFS	۹۴/۱۲	۸۲/۰۵	۸۷/۶۷*	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
DT	All	۸۹/۴۷	۸۷/۱۸	۸۸/۳۱**	۹۳/۷۵	۹۶/۷۷	۹۵/۲۴
	Chen et al.	۹۰/۲۴	۹۴/۸۷	۹۲/۵۰**	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	CFS	۸۹/۷۴	۸۹/۷۴	۸۹/۷۴**	۹۶/۷۷	۹۶/۷۷	۹۶/۷۷
	LRFS	۹۰/۲۴	۹۴/۸۷	۹۲/۵۰**	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
GBT	All	۸۶/۸۴	۸۴/۶۲	۸۵/۷۱*	۹۶/۷۷	۹۶/۷۷	۹۶/۷۷*
	Chen et al.	۸۹/۷۴	۸۹/۷۴	۸۹/۷۴	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	CFS	۸۶/۴۹	۸۲/۰۵	۸۴/۲۱	۹۶/۷۷	۹۶/۷۷	۹۶/۷۷
	LRFS	۸۹/۷۴	۸۹/۷۴	۸۹/۷۴	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
RF	All	۸۶/۸۴	۸۴/۶۲	۸۵/۷۱*	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	Chen et al.	۸۹/۷۴	۸۹/۷۴	۸۹/۷۴*	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	CFS	۹۱/۸۹	۸۷/۱۸	۸۹/۴۷*	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱**
	LRFS	۸۹/۷۴	۸۹/۷۴	۸۹/۷۴*	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
NB	All	۵۶/۸۶	۷۴/۳۶	۶۴/۴۴	۵۷/۴۵	۸۷/۱۰	۶۹/۲۳
	Chen et al.	۹۴/۲۹	۸۴/۶۲	۸۹/۱۹	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	CFS	۸۱/۴۰	۸۹/۷۴	۸۵/۳۷	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	LRFS	۹۴/۲۹	۸۴/۶۲	۸۹/۱۹	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
MLP	All	۸۱/۲۵	۶۶/۶۷	۷۳/۲۳**	۱۰۰/۰۰	۳۵/۴۸	۵۲/۳۸**
	Chen et al.	۹۴/۲۹	۸۴/۶۲	۸۹/۱۹*	۱۰۰/۰۰	۳۵/۴۸	۵۲/۳۸**
	CFS	۸۴/۸۵	۷۱/۷۹	۷۷/۷۸*	۱۰۰/۰۰	۲۹/۰۳	۴۵/۰۰*
	LRFS	۹۴/۲۹	۸۴/۶۲	۸۹/۱۹*	۱۰۰/۰۰	۳۵/۴۸	۵۲/۳۸**
SVM	All	۸۶/۲۱	۶۴/۱۰	۷۳/۵۳**	۰/۰۰	۰/۰۰	۰/۰۰
	Chen et al.	۹۴/۴۴	۸۷/۱۸	۹۰/۶۷	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱
	CFS	۷۹/۳۱	۵۸/۹۷	۶۷/۶۵*	۱۰۰/۰۰	۳۲/۲۶	۴۸/۷۸**
	LRFS	۹۴/۴۴	۸۷/۱۸	۹۰/۶۷	۹۶/۸۸	۱۰۰/۰۰	۹۸/۴۱

جدول ۷- ویژگی‌های انتخاب‌شده در پایان مراحل انتخاب ویژگی برای هر کد مشکوک

	بدون پیش‌پردازش	پیش‌پردازش بدون استفاده از روش SMOTE	پیش‌پردازش با استفاده از روش SMOTE
LM	CountStmtExe CountLineBlank Cyclomatic CountStmt CountLineCodeDecl CountLineCode	CountLineCodeExe CountStmtExe CountLineCode CountPathLog	CountStmtExe Cyclomatic MLOC CyclomaticStrict
LPL	CountStmt CountStmtDecl CountStmtExe number of parameter CountLineCodeDecl	CountLineBlank CountLineCode number of parameter without *args, **kargs CountLine CountLineCodeExe CountLineComment PAR CountStmt	number of parameter CountLineCodeExe CountLineCode CountLineBlank PAR number of parameter without *args, **kargs CountStmt
LSC	CountStmtDecl CountLineCode DOC CountLineCodeDecl CountLineCodeExe	CountLineCodeExe CountLineCode DOC CountStmtDecl	CountStmt DOC CountStmtDecl CountLineBlank CountLineComment
LC	CountClassCoupled CountLineCode CountDeclInstanceMethod CountDeclMethod SumCyclomaticModified SumCyclomatic CountLineCodeDecl CLOC	SumCyclomaticStrict CountStmt CLOC SumCyclomatic SumCyclomaticModified CountStmtExe CountLineComment	CountDeclMethod CLOC CountLineCodeExe CountClassCoupled SumCyclomaticModified SumCyclomatic
LBCL	SumCyclomatic CountLineCodeDecl NBC CountClassBase SumEssential	SumCyclomatic AvgLineComment NBC CountLineBlank	NBC SumEssential CountClassBase SumCyclomaticStrict

جدول ۸- نتایج ارزیابی مجموعه‌داده مربوط به هر کد مشکوک (بدون پیش‌پردازش و مراحل انتخاب ویژگی)

کد مشکوک / معیار ارزیابی	LM	LPL	LSC	LC	LBCL
ACC. (%)	۹۷/۵۰	۸۳/۳۳	۸۰/۵۸	۹۳/۳۳	۹۱/۱۱
Precis. (%)	۹۷/۷۵	۷۰/۷۳	۷۲/۹۷	۹۲/۵۲	۸۷/۴۱
Recall (%)	۹۷/۵۰	۸۳/۳۳	۸۰/۵۸	۹۳/۳۳	۹۱/۱۱
F-measure (%)	۹۷/۴۹	۷۶/۹۶	۷۵/۵۰	۹۲/۳۴	۸۹/۱۰
MCC (%)	۸۲/۸۴	۱۰/۹۹	۱۰/۸۲	۵۸/۹۹	۲۲/۲۹

همان‌طور که در جدول (۹) مشاهده می‌شود، خروجی روش پیشنهادی ارائه‌شده در این مقاله، در دو حالت پیش‌پردازش بدون استفاده از روش SMOTE و پیش‌پردازش با استفاده از روش SMOTE بهبود قابل‌توجهی داشته است. حتی در حالت بدون پیش‌پردازش نیز در سه کد مشکوک متد طولانی، کلاس بزرگ و لیست طولانی از کلاس‌های پایه، در ۳ معیار صحت، پوشش و معیار F بهبود داشته است. معیار MCC نیز یک معیار نسبتاً جدید است و یک معیار ایده‌آل برای مجموعه‌داده‌های نامتعادل در نظر گرفته می‌شود. معیار MCC، مقادیر امتیاز را بین ۱ و ۱- برمی‌گرداند که در آن ۱ نشان‌دهنده یک مدل کامل، ۱- نشان‌دهنده طبقه‌بندی اشتباه کامل و ۰ نشان‌دهنده پیش‌بینی تصادفی است. بر این اساس، همانطور که مشاهده می‌شود، این معیار برای روش ارائه‌شده در این مقاله مقادیری نزدیک به ۱ را محاسبه کرده است و این نشان از عملکرد بالای روش پیشنهادی در شناسایی کدهای مشکوک مورد بحث دارد.

معیار دقت، تنها احتمال پیش‌بینی صحیح ماژول‌های معیوب و غیر معیوب را برای مدل طبقه‌بند نشان می‌دهد و هیچ اطلاعاتی در مورد پیش‌بینی نادرست آنها، ارائه نمی‌دهد؛ لذا، دستیابی به عملکرد بالای این معیار، به‌تنهایی نمی‌تواند نشان‌دهنده میزان کارایی روش پیشنهادی باشد و لازم است ۴ معیار ارزیابی در کنار هم برای ارزیابی عملکرد نهایی، مورد بررسی قرار گیرند. همان‌طور که نتایج نشان داد و در مقایسه با نتایج دو تحقیق واتاناپاکورن و همکاران و همچنین چن و همکاران، روش پیشنهادی در این مقاله توانست در تمامی معیارها، امتیاز بالایی را به دست آورد که نشان از کارایی بالای آن، در تشخیص ۵ کد مشکوک مورد بحث دارد.

در این مقاله، به اهمیت پیش‌بینی کدهای مشکوک و نقش آن در نگهداری و بهبود کیفیت نرم‌افزار پرداخته شده است. این موضوع در صنعت نرم‌افزار بسیار حائز اهمیت است؛ زیرا می‌تواند باعث کاهش هزینه‌ها و افزایش رضایتمندی کاربران شود. همچنین، استفاده از معیارهای ارزیابی مختلف مانند دقت، صحت، پوشش و معیار MCC، باعث شده که تحلیل نتایج جامع‌تر گردد. این معیارها، قابلیت اطمینان و کاربردی‌بودن مدل را نشان می‌دهند.

جدول ۹- مقایسه نتایج روش پیشنهادی در این مقاله (با اعمال مراحل انتخاب ویژگی) با روش واتاناپاکورن و همکاران و کارچن و همکاران

		کد مشکوک / معیار	LM	LPL	LSC	LC	LBCL
چن و همکاران		الگوریتم	TM	TM	TM	TM	TM
		ویژگی	MLOC	PAR	DOC	CLOC	NBC
		ACC. (%)	۹۸/۸۹	۹۶/۱۱	۹۴/۵۷	۹۸/۳۳	۹۹/۷۲
		Precis. (%)	۸۹/۶۶	۹۶/۲۳	۱۰۰/۰۰	۹۰/۲۴	۹۶/۸۸
		Recall (%)	۹۶/۳	۸۰/۹۵	۷۰/۸۳	۹۴/۸۷	۱۰۰/۰۰
		F-measure (%)	۹۲/۸۶	۸۷/۹۳	۸۲/۹۳	۹۲/۵	۹۸/۴۱
واتاناپاکورن و همکاران		بهترین الگوریتم	DT GBT RF	RF	DT GBT RF	K-NN	K-NN LR DT GBT RF NB SVM
		ویژگی	MLOC	PAR NP	DOC CountLineCodeDecl	CLOC	NBC
		ACC. (%)	۹۹/۷۲	۹۷/۲۲	۹۹/۲۲	۹۸/۶۱	۹۹/۷۲
		Precis. (%)	۹۶/۴۳	۹۲/۰۶	۱۰۰/۰۰	۹۲/۵۰	۹۶/۸۸
		Recall (%)	۱۰۰/۰۰	۹۲/۰۶	۹۵/۸۳	۹۴/۸۷	۱۰۰/۰۰
		F-measure (%)	۹۸/۱۸	۹۲/۰۶	۹۷/۸۷	۹۳/۶۷	۹۸/۴۱
نتایج ارزیابی روش پیشنهادی	بدون پیش‌پردازش (٪)	ACC.	۹۹/۷۲	۹۲/۵۰	۹۵/۳۸	۹۸/۳۳	۹۹/۷۲
		Precis.	۹۹/۷۳	۹۳/۸۲	۹۶/۰۹	۹۸/۴۳	۹۹/۷۹
		Recall	۹۹/۷۲	۹۲/۵۰	۹۵/۳۸	۹۸/۳۳	۹۹/۷۲
		F-measure	۹۹/۷۰	۹۲/۸۴	۹۴/۷۹	۹۸/۲۵	۹۹/۷۴
		MCC	۹۸/۰۴	۷۷/۴۵	۸۳/۳۷	۹۰/۹۶	۹۸/۵۳
	پیش‌پردازش بدون استفاده از روش SMOTE (٪)	ACC.	۹۹/۷۲	۹۷/۲۲	۹۸/۴۶	۹۹/۱۷	۹۹/۷۲
		Precis.	۹۹/۷۹	۹۷/۳۱	۹۸/۵۹	۹۹/۲۹	۹۹/۷۹
		Recall	۹۹/۷۲	۹۷/۲۲	۹۸/۴۶	۹۹/۱۷	۹۹/۷۲
		F-measure	۹۹/۷۴	۹۷/۱۲	۹۸/۲۴	۹۹/۱۸	۹۹/۷۴
		MCC	۹۸/۵۳	۹۰/۰۴	۹۳/۵۴	۹۶/۱۳	۹۸/۵۳
	پیش‌پردازش با استفاده از روش SMOTE (٪)	ACC.	۹۹/۷۷	۹۸/۲۳	۹۸/۵۷	۹۹/۳۰	۹۹/۷۷
		Precis.	۹۹/۷۹	۹۸/۲۹	۹۸/۷۷	۹۹/۳۴	۹۹/۷۹
		Recall	۹۹/۷۷	۹۸/۲۳	۹۸/۵۷	۹۹/۳۰	۹۹/۷۷
		F-measure	۹۹/۷۸	۹۸/۲۲	۹۸/۵۴	۹۹/۳۰	۹۹/۷۷
		MCC	۹۹/۴۳	۹۵/۲۶	۹۶/۳۰	۹۸/۱۵	۹۹/۳۹

در این مقاله پنج نوع کد مشکوک رایج (مانند متد طولانی و کلاس بزرگ) بررسی شده که تمرکز بر چالش‌های رایج در طراحی و نگهداری نرم‌افزارها است و مرتبط با نیازهای صنعتی است. در مواردی نظیر سامانه‌های کنترل پالایشگاه‌ها، کنترل ترافیک هوایی[۱]، شبیه‌سازی پرواز هواپیما و غیره که معمولاً حجم کد این سامانه‌ها از ۱/۵ میلیون خط برنامه فراتر هستند، وجود کدهای مشکوک، نگهداری و توسعه آنها را به‌شدت با چالش مواجه می‌سازد.

۶. نتیجه‌گیری

وقوع اشکال در نرم‌افزار یک اتفاق رایج محسوب می‌شود، اما شناسایی زودهنگام اشکال برای توسعه‌دهندگان نرم‌افزار اهمیت بالایی دارد؛ زیرا هر اندازه که خطاها در نرم‌افزار دیرتر شناسایی شوند، تلاش و هزینه بیشتری برای رفع آنها نیاز خواهد بود. پیش‌بینی خطاهای نرم‌افزار به معنای شناسایی کدهای مستعد خطا در مراحل اولیه توسعه نرم‌افزار است. کد مشکوک یک نشانه سطحی است که معمولاً مربوط به یک مشکل عمیق‌تر در سیستم است. کد مشکوک، نگهداری، توسعه و تکامل برنامه را با مشکل مواجه می‌کند. در این مقاله، یک روش انتخاب ویژگی جدید با استفاده از روش‌های انتخاب ویژگی مجموعه‌ای (شامل بهره اطلاعاتی، امتیاز بهره اطلاعاتی و کای-مربع) و فنون پوششی (شامل الگوریتم ژنتیک و بهینه‌ساز گرگ خاکستری) ارائه گردید. همچنین از شبکه عصبی پرسپترون چندلایه پس انتشار خطا به‌عنوان طبقه‌بند بهره گرفته شد. برای ارزیابی این روش، از اعتبارسنجی متقابل ۱۰-بخشی طبقه‌بندی‌شده استفاده گردید. در روش پیشنهادی، از ترکیب الگوریتم‌های ژنتیک و گرگ خاکستری استفاده شده که می‌تواند دقت و کارایی در انتخاب ویژگی را بهبود بخشد. این روش برای حل مشکلات بهینه‌سازی و کاهش زمان آموزش مدل‌ها مناسب بوده و می‌تواند کارایی بالاتری در شناسایی کدهای مشکوک ارائه دهد. همان‌طور که در بخش پنجم گفته شد، نتایج نشان می‌دهد که روش پیشنهادی قادر است با انتخاب ویژگی‌های بهینه و با استفاده از طبقه‌بند مورد نظر (پرسپترون چندلایه پس انتشار خطا) نتایج مطلوبی را محاسبه نماید. روش پیشنهادی توانست ۴ معیار ارزیابی دقت، صحت، پوشش و معیار F را برای ۵ کد مشکوک متد طولانی، لیست پارامترهای طولانی، زنجیره دامنه طولانی، کلاس بزرگ و لیست طولانی از کلاس‌های پایه، بهبود دهد. همچنین معیار ارزیابی MCC نزدیک به ۱ را که نشان‌دهنده یک مدل کامل است، برای کدهای مشکوک به دست آورد.

استفاده از معیارهای ارزیابی مختلف مانند دقت، صحت، پوشش و معیار MCC، تحلیل نتایج را جامع‌تر کرده است. این معیارها، قابلیت اطمینان و کاربردی‌بودن مدل را نشان می‌دهند. هدف نهایی و اصلی این مقاله که بهبود کیفیت نرم‌افزار به کمک پیش‌بینی زودهنگام کدهای مشکوک در کد منبع نرم‌افزار با استفاده از یک روش انتخاب ویژگی مطلوب است، با روش پیشنهادی تحقق پیدا کرد. به‌تبع آن، کاهش هزینه‌های آزمون، بالابردن خوانایی و قابلیت نگهداری نرم‌افزار و جلوگیری از شکست احتمالی در آینده نیز محقق گردید.

به‌عنوان پیشنهادی برای کارهای آینده می‌توان موارد زیر را مدنظر قرارداد:

۱. همان‌طور که گفته شد، فنون انتخاب ویژگی مختلفی وجود دارد. در این مقاله از روش‌های انتخاب ویژگی مجموعه‌ای و فنون پوششی بهره گرفته شد که در این راستا می‌توان از فنون دیگری مانند فن تعبیه‌شده استفاده کرد و زمان مرحله انتخاب ویژگی را مورد بررسی قرارداد.
۲. مجموعه‌داده‌ای که برای این تحقیق استفاده شده است کوچک است. تولید مجموعه‌داده بزرگ‌تر که بتوان روی آن مدل را آموزش داد، می‌تواند به غنی‌تر کردن کار بسیار کمک کند.

۷. مراجع

[۱] Rathore, S. S.; Kumar, S. "A Study on Software Fault Prediction Techniques"; Artif. Intell. Rev. ۲۰۱۹, ۵۱, ۲۵۵-۳۲۷. doi: ۱۰.۱۰۰۷/s۱۰۴۶۲-۰۱۷-۹۵۶۳-۵.

[۲] Jain, S.; Saha, A. "Improving Performance with Hybrid Feature Selection and Ensemble Machine Learning Techniques for Code Smell Detection"; Sci. Comput. Program. ۲۰۲۱, ۲۱۲, ۱۰۲۷۱۳. doi: ۱۰.۱۰۱۶/j.scico.۲۰۲۱.۱۰۲۷۱۳.

[۳] Tufano, M.; Palomba, F.; Bavota, G.; Oliveto, R.; Di Penta, M.; De Lucia, A.; Poshyvanyk, D. "When and Why Your Code Starts to Smell Bad"; ۳۷th IEEE Int. Conf. Softw. Eng. ۲۰۱۵, ۱, ۴۰۳-۴۱۴. doi: ۱۰.۱۱۰۹/ICSE.۲۰۱۵.۵۹.

[۴] Gupta, A.; Suri, B.; Misra, S. "A Systematic Literature Review: Code Bad Smells in Java Source Code"; Int. Conf. Comput. Sci. Appl. ۲۰۱۷, ۶۶۵-۶۸۲. doi: ۱۰.۱۰۰۷/۹۷۸-۳-۳۱۹-۶۲۴۰۴-۴_۴۹.

[۵] Mahalakshmi, D.; Kasinathan, P.; Elangovan, D.; Bhat, C. R.; Balamurugan, M.; Sivakumar, S. "Code Smell Detection using Hybrid Machine Learning Algorithms"; ۵th Int. Conf. Inventive Res. Comput. Appl. ۲۰۲۳, ۶۳۳-۶۳۸. doi: ۱۰.۱۱۰۹/ ICIRCA۵۷۹۸۰.۲۰۲۳.۱۰۲۲۰۹۱۱.

[۶] Zhang, Y.; Ge, C.; Liu, H.; Zheng, K. "Code Smell Detection Based on Supervised Learning Models: A Survey"; Neurocomputing ۲۰۲۴, ۵۶۵, ۱۲۷۰۱۴. https://doi.org/۱۰.۱۰۱۶/ j.neucom.۲۰۲۳.۱۲۷۰۱۴.

[۷] Olbrich, S. M.; Cruzes, D. S.; Sjøberg, D. I. "Are All Code Smells Harmful? A study of God Classes and Brain Classes in the Evolution of Three Open Source Systems"; IEEE Int. Conf. Softw. Maint. ۲۰۱۰, ۱-۱۰. doi: ۱۰.۱۱۰۹/ICSM.۲۰۱۰.۵۶۰۹۵۶۴.

[۸] Moha, N.; Guéhéneuc, Y.-G.; Duchien, L.; Le Meur, A.-F. "Decor: A Method for the Specification and Detection of Code and Design Smells"; IEEE Trans. Softw. Eng. ۲۰۰۹, ۳۶, ۲۰-۳۶. doi: ۱۰.۱۱۰۹/TSE.۲۰۰۹.۵۰.

۱. Air Traffic Control (ATC)

[۹] Zhang, X.-F.; Zhu, C. “Empirical Study of Code Smell Impact on Software Evolution”; J. Softw. ۲۰۱۹, ۳۰, ۱۴۲۲-۱۴۳۷. doi: ۱۰.۱۳۳۲۸/j.cnki.jos.۰۰۵۷۳۵.

[۱۰] Tian, Y. C.; Li, K. J.; Wang, T. M.; Jiao, Q. Q.; Li, G. J.; Zhang, Y. X.; Liu, H. “Survey on Code Smells. Ruan Jian Xue Bao”; J. Softw. ۲۰۲۳, ۳۴, ۱۵۰-۱۷۰. doi: ۱۰.۱۳۳۲۸/j.cnki.jos.۰۰۶۴۳۱.

[۱۱] Azeem, M. I.; Palomba, F.; Shi, L.; Wang, Q. “Machine Learning Techniques for Code Smell Detection: A Systematic Literature Review and Meta-Analysis”; Inf. Softw. Technol. ۲۰۱۹, ۱۰۸, ۱۱۵-۱۳۸. doi: ۱۰.۱۰۱۶/j.infsof.۲۰۱۸.۱۲.۰۰۹.

[۱۲] Caram, F. L.; Rodrigues, B. R. D. O.; Campanelli, A. S.; Parreiras, F. S. “Machine Learning Techniques for Code Smells Detection: A Systematic Mapping Study”; Int. J. Softw. Eng. Knowl. Eng. ۲۰۱۹, ۲۹, ۲۸۵-۳۱۶. doi: ۱۰.۱۱۴۲/S۰۲۱۸۱۹۴۰۱۹۵۰۰۱۳X.

[۱۳] Pereira dos Reis, J.; Brito e Abreu, F.; de Figueiredo Carneiro, G.; Anslow, C. “Code Smells Detection and Visualization: A Systematic Literature Review”; Arch. Comput. Methods Eng. ۲۰۲۲, ۲۹, ۴۷-۹۴. doi: ۱۰.۱۰۰۷/s۱۱۸۳۱-۰۲۱-۰۹۵۶۶-x.

[۱۴] Vatanapakorn, N.; Soomlek, C.; Seresangtakul, P. “Python Code Smell Detection Using Machine Learning”; ۲۶th Int. Comput. Sci. Eng. Conf. (ICSEC) ۲۰۲۲, ۱۲۸-۱۳۳. doi: ۱۰.۱۱۰۹/ICSEC۵۶۳۳۷.۲۰۲۲.۱۰۰۴۹۳۳۰.

[۱۵] Chen, Z.; Chen, L.; Ma, W.; Zhou, X.; Zhou, Y.; Xu, B. “Understanding Metric-Based Detectable Smells in Python Software: A Comparative Study”; Inf. Softw. Technol. ۲۰۱۸, ۹۴, ۱۴-۲۹. doi: ۱۰.۱۰۱۶/j.infsof.۲۰۱۷.۰۹.۰۱۱.

[۱۶] Abdou, A.; Darwish, N. “Severity Classification of Software Code Smells Using Machine Learning Techniques: A comparative study”; J. Softw. ۲۰۲۴, ۳۶, e۲۴۵۴. doi: ۱۰.۱۰۰۲/smr.۲۴۵۴.

[۱۷] Sandouka, R.; Aljamaan, H. “Python Code Smells Detection Using Conventional Machine Learning Models”; PeerJ Comput. Sci. ۲۰۲۳, ۹, e۱۳۷۰. doi: ۱۰.۷۷۱۷/peerj-cs.۱۳۷۰.

[۱۸] Jain, S.; Saha, A. “Rank-Based Univariate Feature Selection Methods on Machine Learning Classifiers for Code Smell Detection”; Evol. Intell. ۲۰۲۲, ۱۵, ۶۰۹-۶۳۸. doi: ۱۰.۱۰۰۷/s۱۲۰۶۵-۰۲۰-۰۰۵۳۶-z.

[۱۹] Alsghaier, H.; Akour, M. “Software Fault Prediction Using Whale Algorithm with Genetics Algorithm”; Softw. Pract. Exp. ۲۰۲۱, ۵۱, ۱۱۲۱-۱۱۴۶. doi: ۱۰.۱۰۰۲/spe.۲۹۴۱.

[۲۰] Fontana, F. A.; Mäntylä, M. V.; Zanoni, M.; Marino, A. “Comparing and Experimenting Machine Learning Techniques for Code Smell Detection”; Empir. Softw. Eng. ۲۰۱۶, ۲۱, ۱۱۴۳-۱۱۹۱. doi: ۱۰.۱۰۰۷/s۱۰۶۶۴-۰۱۵-۹۳۷۸-۴.

[۲۱] Zhu, Z.; Ong, Y.-S.; Dash, M. “Wrapper–Filter Feature Selection Algorithm Using a Memetic Framework”; IEEE Trans. Syst. Man Cybern. B ۲۰۰۷, ۳۷, ۷۰-۷۶. doi: ۱۰.۱۱۰۹/TSMCB.۲۰۰۶.۸۸۳۲۶۷.

[۲۲] Riaz, S.; Arshad, A.; Jiao, L. “Rough Noise-Filtered Easy Ensemble for Software Fault Prediction”; IEEE Access ۲۰۱۸, ۶, ۴۶۸۸۶-۴۶۸۹۹. doi: ۱۰.۱۱۰۹/ACCESS.۲۰۱۸.۲۸۶۵۳۸۳.

[۲۳] Gao, K.; Khoshgoftaar, T. M.; Wang, H.; Seliya, N. “Choosing Software Metrics for Defect Prediction: An Investigation on Feature Selection Techniques”; Softw. Pract. Exp. ۲۰۱۱, ۴۱, ۵۷۹-۶۰۶ doi: ۱۰.۱۰۰۲/spe.۱۰۴۳.

[۲۴] Kalsoom, A.; Maqsood, M.; Ghazanfar, M. A.; Aadil, F.; Rho, S. “A Dimensionality Reduction-Based Efficient Software Fault Prediction Using Fisher Linear Discriminant Analysis (FLDA)”; J. Supercomput. ۲۰۱۸, ۷۴, ۴۵۶۸-۴۶۰۲. doi: ۱۰.۱۰۰۷/s۱۱۲۲۷-۰۱۸-۲۳۲۶-۵.

[۲۵] Alazba, A.; Aljamaan, H.; Alshayeb, M. “Deep Learning Approaches for Bad Smell Detection: A Systematic Literature Review”; Empir. Softw. Eng. ۲۰۲۳, ۲۸, ۷۷. doi: ۱۰.۱۰۰۷/s۱۰۶۶۴-۰۲۳-۱۰۳۱۲-z.

[۲۶] Bolón-Canedo, V.; Sánchez-Marono, N.; Alonso-Betanzos, A.; Benítez, J. M.; Herrera, F. “A Review of Microarray Datasets and Applied Feature Selection Methods”; Inf. Sci. ۲۰۱۴, ۲۸۲, ۱۱۱-۱۳۵. doi: ۱۰.۱۰۱۶/j.ins.۲۰۱۴.۰۵.۰۴۲.

[۲۷] Almazrua, H.; Alshamlan, H. “A Comprehensive Survey Of Recent Hybrid Feature Selection Methods in Cancer Microarray Gene Expression Data”; IEEE Access ۲۰۲۲, ۱۰, ۷۱۴۲۷-۷۱۴۴۹. doi: ۱۰.۱۱۰۹/ACCESS.۲۰۲۲.۳۱۸۵۲۲۶.

[۲۸] Remeseiro, B.; Bolon-Canedo, V. “A Review of Feature Selection Methods in Medical Applications”; Comput. Biol. Med. ۲۰۱۹, ۱۱۲, ۱۰۳۳۷۵. doi: ۱۰.۱۰۱۶/j.compbiomed. ۲۰۱۹. ۱۰۳۳۷۵.

[۲۹] De, R.; Bush, W. S.; Moore, J. H. “Bioinformatics Challenges in Genome-Wide Association Studies (GWAS)”; Clin. Bioinform. ۲۰۱۴, ۶۳-۸۱. doi: ۱۰.۱۰۰۷/۹۷۸-۱-۴۹۳۹-۰۸۴۷-۹_۵.

[۳۰] Okser, S.; Pahikkala, T.; Aittokallio, T. “Genetic Variants and Their Interactions in Disease Risk Prediction–Machine Learning and Network Perspectives”; BioData Min. ۲۰۱۳, ۶, ۱-۱۶. doi: ۱۰.۱۱۸۶/۱۷۵۶-۰۳۸۱-۶-۵.

[۳۱] Saeys, Y.; Inza, I.; Larranaga, P. “A Review of Feature Selection Techniques in Bioinformatics”; Bioinform. ۲۰۰۷, ۲۳, ۲۵۰۷-۲۵۱۷. doi: ۱۰.۱۰۹۳/bioinformatics/btm۳۴۴.

[۳۲] Pudjihartono, N.; Fadason, T.; Kempa-Liehr, A. W.; O'Sullivan, J. M. “A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction”; Front. Bioinform. ۲۰۲۲, ۲, ۹۲۷۳۱۲. doi: ۱۰.۳۳۸۹/fbinf.۲۰۲۲.۹۲۷۳۱۲.

[۳۳] Jain, D.; Singh, V. “Feature Selection and Classification Systems for Chronic Disease Prediction: A Review”; Egypt. Inform. J. ۲۰۱۸, ۱۹, ۱۷۹-۱۸۹. doi: ۱۰.۱۰۱۶/j.eij.۲۰۱۸.۰۳.۰۰۲.

[۳۴] Chen, C. W.; Tsai, Y. H.; Chang, F. R.; Lin, W. C. “Ensemble Feature Selection in Medical Datasets: Combining Filter, Wrapper, and Embedded Feature Selection Results”; Expert Syst. ۲۰۲۰, ۳۷, e۱۲۵۵۳. doi: ۱۰.۱۱۱۱/exsy.۱۲۵۵۳.

[۳۵] Balogun, A. O.; Basri, S.; Abdulkadir, S. J.; Hashim, A. S. “Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach”; Appl. Sci. ۲۰۱۹, ۹, ۲۷۶۴. doi: ۱۰.۳۳۹۰/app۹۱۳۲۷۶۴.

[۳۶] Mafarja, M.; Qasem, A.; Heidari, A. A.; Aljarah, I.; Faris, H.; Mirjalili, S. “Efficient Hybrid Nature-Inspired Binary Optimizers for Feature Selection”; Cognit. Comput. ۲۰۲۰, ۱۲, ۱۵۰-۱۷۵. doi: ۱۰.۱۰۰۷/s۱۲۵۵۹-۰۱۹-۰۹۶۶۸-۶.

[۳۷] Mohammed, H. M.; Umar, S. U.; Rashid, T. A. “A Systematic and Meta-Analysis Survey of Whale Optimization

Algorithm"; Comput. Intell. Neurosci. ۲۰۱۹, ۲۰۱۹, ۸۷۱۸۵۷۱. doi: ۱۰.۱۱۵۵/۲۰۱۹/۸۷۱۸۵۷۱.

[۳۸] Too, J.; Abdullah, A. R. "A New and Fast Rival Genetic Algorithm for Feature Selection"; J. Supercomput. ۲۰۲۱, ۷۷, ۲۸۴۴-۲۸۷۴. doi: ۱۰.۱۰۰۷/s۱۱۲۲۷-۰۲۰-۰۳۳۷۸-۹.

[۳۹] Keshanchi, B.; Souri, A.; Navimipour, N. J. "An Improved Genetic Algorithm for Task Scheduling in the Cloud Environments Using the Priority Queues: Formal Verification, Simulation, and Statistical Testing"; J. Syst. Softw. ۲۰۱۷, ۱۲۴, ۱-۲۱. doi: ۱۰.۱۰۱۶/j.jss.۲۰۱۶.۰۷.۰۰۶.

[۴۰] Turabieh, H.; Mafarja, M.; Li, X. "Iterated Feature Selection Algorithms with Layered Recurrent Neural Network for Software Fault Prediction"; Expert. Syst. Appl. ۲۰۱۹, ۱۲۲, ۲۷-۴۲. doi: ۱۰.۱۰۱۶/j.eswa.۲۰۱۸.۱۲.۰۳۳.

[۴۱] Sadeghian, Z.; Akbari, E.; Nematzadeh, H.; Motameni, H. "A Review of Feature Selection Methods Based on Meta-Heuristic Algorithms"; J. Exp. Theor. Artif. Intell. ۲۰۲۳, ۱-۵۱. doi: ۱۰.۱۰۸۰/۰۹۵۲۸۱۳X.۲۰۲۳.۲۱۸۳۲۶۷

[۴۲] Emary, E.; Zawbaa, H. M.; Hassanien, A. E. "Binary Grey Wolf Optimization Approaches for Feature Selection"; Neurocomputing ۲۰۱۶, ۱۷۲, ۳۷۱-۳۸۱. doi: ۱۰.۱۰۱۶/j.neucom. ۲۰۱۵.۰۶.۰۸۳

[۴۳] Chowdhury, A. A.; Borkar, V. S.; Birajdar, G. K. "Indian Language Identification Using Time-Frequency Image Textural Descriptors and GWO-Based Feature Selection"; J. Exp. Theor. Artif. Intell. ۲۰۲۰, ۳۲, ۱۱۱-۱۳۲. doi: ۱۰.۱۰۸۰/۰۹۵۲۸۱۳X. ۲۰۱۹.۱۶۳۱۳۹۲.

[۴۴] Pramanik, R.; Pramanik, P.; Sarkar, R. "Breast Cancer Detection in Thermograms Using a Hybrid of GA and GWO Based Deep Feature Selection Method"; Expert Syst. Appl. ۲۰۲۳, ۲۱۹, ۱۱۹۶۴۳. doi: ۱۰.۱۰۱۶/j.eswa.۲۰۲۳.۱۱۹۶۴۳.

[۴۵] Mirjalili, S.; Mirjalili, S. M.; Lewis, A. "Grey Wolf Optimizer"; Adv. Eng. Softw. ۲۰۱۴, ۶۹, ۴۶-۶۱. doi: ۱۰.۱۰۱۶/ j.advengsoft.۲۰۱۳.۱۲.۰۰۷.

[۴۶] Abbas, A. K.; Al-haideri, N. A.; Bashikh, A. A. "Implementing Artificial Neural Networks and Support Vector Machines to Predict Lost Circulation"; Egypt. J. Pet. ۲۰۱۹, ۲۸, ۳۳۹-۳۴۷. doi: ۱۰.۱۰۱۶/j.ejpe.۲۰۱۹.۰۶.۰۰۶.

[۴۷] Ghatak, A. "Introduction to Machine Learning"; Mach. Learn. R ۲۰۱۷, ۵۷-۷۸. doi: ۱۰.۱۰۰۷/۹۷۸-۹۸۱-۱۰-۶۸۰۸-۹_۳.

[۴۸] Mutasa, S.; Sun, S.; Ha, R. "Understanding Artificial Intelligence Based Radiology Studies: What Is Overfitting?"; Clin. Imaging ۲۰۲۰, ۶۵, ۹۶-۹۹. doi: ۱۰.۱۰۱۶/ j.clinimag.۲۰۲۰.۰۴.۰۲۵.