Electronic & Cyber Defense
Vol. 11, No. 3, Autumn 2023, Serial No. 43
ISSN: 2322-4347 , E-ISSN: 2980-8979

A method to prediction of software system's code smells using neural network

A. Karimi[*1], F. karimi[2]

assistant professor, Imam Hossein University, Tehran, Iran

((Received: 2023/02/22, Revised: 2023/07/12, Accepted: 2023/07/31 Published: 2023/09/28))

DOR: https://dorl.net/dor/20.1001.1.23224347.1402.11.3.7.2

Abstract

Software engineers are always looking to reduce production costs and increase software quality. There are various methods to improve software quality, and code refactoring is one of these methods. Code refactoring and reorganization is a method for cleaning up software code and is one of the crucial processes in maintaining software quality. One of the main challenges in developing and producing clean code in software is the existence of inconsistent or bad-smelling code. Code smell is a superficial sign in the code that may indicate a deeper problem in the software. The existence of code smells may slow down processing, increase the risk of failure, as well as software errors. Therefore, software developers attempt to identify inconsistent code and facilitate its maintainability and scalability by refactoring software code. However, manual and automatic identification of code smells is challenging and tiring. As a result, methods for identifying such codes automatically and semi-automatically have been proposed. An important note in non-automatic methods is that predicting inconsistent code requires individual knowledge that is both time-consuming and increases the possibility of error. Therefore, automated methods have a greater advantage in predicting inconsistent code. So far, extensive research has been conducted on automatic prediction and identification of inconsistent code. A high percentage of these studies have focused on predicting four types of code smells: long method, feature envy, god class, and data class. In this article, our focus is on improving the accuracy of extracting such inconsistent codes. One of the common methods for predicting this type of code is using machine learning-based methods. Artificial neural networks are a specific type of machine learning algorithm that is modeled according to the human brain's performance method. This means that these networks can learn from input data and provide responses in the form of predictions and classifications. In this article, a multi-layer neural network was used to predict software inconsistent code, as well as a new feature selection method to increase prediction accuracy.

Keywords: code smell, feature selection, classification, machine learning, neural network

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

"پدافند الکترونیکی و سایبری"

سال یازدهم، شماره ۳، پاییز ۱۴۰۲، ص ۷۶-۶۷

ارائه روشی برای پیش‌بینی کدهای نابسامان سامانه‌های نرم‌افزاری با استفاده از شبکه عصبی

علی کریمی[1*]، فرهاد کریمی[2]

۱- استادیار، ۲- دانشجوی کارشناسی ارشد، دانشگاه جامع امام حسین (ع)، تهران، ایران

(دریافت: ۱۴۰۱/۱۲/۰۳، بازنگری: ۱۴۰۲/۰۴/۲۱، پذیرش: ۱۴۰۲/۰۵/۰۹، انتشار: ۱۴۰۲/۰۷/۰۶)

DOR: https://dorl.net/dor/20.1001.1.23224347.1402.11.3.7.2

چکیده

مهندسان نرم‌افزار همواره به دنبال کاهش هزینه‌های تولید و افزایش کیفیت نرم‌افزار هستند. روش‌های مختلفی برای افزایش کیفیت نرم‌افزار وجود دارد که بازسازی کد یکی از این روش‌ها است. بازسازی و بازآرایی کد روشی برای تمیز کردن کدهای نرم‌افزار و یکی از روال‌های بسیار مهم در حفظ کیفیت نرم‌افزار است. یکی از چالش‌های اصلی در توسعه و تولید کدهای تمیز در نرم‌افزار وجود کدهای نابسامان یا بوهای کد است. بوی کد یک نشانه سطحی در کد است که احتمالاً نشان‌دهنده‌ی یک مشکل عمیق‌تر در نرم‌افزار می‌باشد. وجود بوی کد ممکن است باعث کند شدن پردازش، افزایش خطر خرابی و همچنین خطاهای نرم‌افزار شود. از این‌رو، توسعه‌دهندگان نرم‌افزار درصدد هستند که با شناسایی کدهای نابسامان، ضمن بازآرایی کد نرم‌افزار، توسعه‌پذیری و نگهداشت‌پذیری آن را در آینده تسهیل کنند. با این حال، شناسایی دستی و غیرخودکار بوهای کد چالش‌برانگیز و خسته‌کننده است. بنابراین، روش‌هایی برای شناسایی این نوع کدها به‌صورت خودکار و نیمه‌خودکار ارایه شده است. نکته حائز اهمیت در روش‌های غیرخودکار آن است که پیش‌بینی کدهای نابسامان، نیاز به دانش فردی افراد است که هم زمان‌بر است و هم امکان خطا را افزایش می‌دهد. ازاین‌رو، استفاده از روش‌های خودکار برای پیش‌بینی کدهای نابسامان، ارجحیت بیشتری دارد. تاکنون تحقیقات زیادی در حوزه پیش‌بینی و شناسایی کدهای نابسامان به‌صورت خودکار انجام شده است. درصد زیادی از این تحقیقات بر روی پیش‌بینی چهار نوع بوی کد شامل؛ متد طولانی، خصیصه حسادت، کلاس خدا و کلاس داده تمرکز کرده‌اند. تمرکز ما نیز در این مقاله بر روی بهبود دقت استخراج این نوع از کدهای نابسامان است. یکی از روش‌های رایج برای پیش‌بینی این نوع کدها، استفاده از روش‌های مبتنی بر یادگیری ماشین است. شبکه‌های عصبی مصنوعی نوع خاصی از الگوریتم‌های یادگیری ماشین است که مطابق با روش عملکرد مغز انسان مدل شده‌اند. به این معنی که این شبکه‌ها قادر هستند از داده‌های ورودی یاد بگیرند و پاسخ را در قالب پیش‌بینی‌ها و طبقه‌بندی‌ها ارائه دهند. در این مقاله، برای پیش‌بینی کدهای نابسامان نرم‌افزار از شبکه عصبی چند لایه و همچنین از یک روش انتخاب ویژگی جدید به‌منظور افزایش دقت پیش‌بینی استفاده شده است.

کلیدواژه‌ها: کد نابسامان، بوی کد، انتخاب ویژگی، طبقه‌بندی، یادگیری ماشین، شبکه عصبی.

۱. مقدمه

امروزه با توجه به رشد نرم‌افزار ازنظر اندازه و پیچیدگی، حفظ کیفیت نرم‌افزار از اهمیت ویژه‌ای برخوردار است. یک روش مؤثر در حفظ کیفیت نرم‌افزار، افزایش کیفیت کد منبع می‌باشد. یکی از راه‌کارهای کلیدی برای افزایش کیفیت کد منبع، شناسایی کدهای نابسامان یا بوهای کد [1] و بازآرایی[2] کد نرم‌افزار است. کدهای نابسامان، نشانه‌های سطحی در کد منبع هستند که ممکن است حاکی از وجود مشکلات عمیق‌تر در سامانه باشند. وقتی در یک سامانه نرم‌افزاری، نشانه‌هایی از کدهای نابسامان مشاهده می‌شود، می‌توان حدس زد که احتمالاً توسعه و نگهداری آن دچار مشکل خواهد شد. کدهای نابسامان، درواقع نشانه‌هایی از زلزله‌ای قریب‌الوقوع در نرم‌افزار هستند. برای تسهیل این شناسایی، مارتین فاولر[3] در سال ۲۰۰۰ مفهوم کدهای نابسامان و همچنین ویژگی‌های هرکدام از این کدها و راه‌حل بالقوه برای برطرف کردن آن‌ها را معرفی کرد [1]. توسعه‌دهندگان می‌توانند با پیش‌بینی کدهای نابسامان در کد نرم‌افزار، قدم بزرگی در افزایش کیفیت کد نرم‌افزار بردارند. با این‌حال، شناسایی دستی کدهای نابسامان چالش‌برانگیز و خسته‌کننده است. ازاین‌رو،

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

[1] Code Smells
[2] Refactoring
[3] Martin Fowler

روش‌های متعددی برای خودکارسازی شناسایی این‌گونه کدها ارایه شده است. استفاده از فنون یادگیری ماشین، یکی از روش‌های خودکار رایج در این زمینه است. روش‌های خودکار مبتنی بر یادگیری ماشین دارای دقت خوبی در استخراج کدهای نابسامان هستند. سؤالی که در اینجا مطرح است این است که آیا می‌توان این دقت را بهبود بخشید؟

پاسخ به این سؤال، مسئله اصلی این مقاله است. در این مقاله، با استفاده از یک روش انتخاب ویژگی جدید به‌منظور انتخاب ویژگی‌های بهینه مجموعه داده و همچنین استفاده از شبکه عصبی به‌عنوان یکی از طبقه‌بندهای رایج در این حوزه، درصدد آن هستیم که دقت پیش‌بینی دو نوع کد نابسامان متداول شامل متدهای طولانی[1] و خصیصه حسادت[2] را بهبود دهیم.

۲. مرور ادبیات تحقیق

در این بخش، موضوعات و مفاهیم مبنایی مرتبط با این مقاله ازجمله مفاهیم طبقه‌بندی، شبکه عصبی، انتخاب ویژگی و الگوریتم‌های مرتبط با آن توضیح داده خواهند شد.

۲-۱. طبقه‌بندی

پرکاربردترین فنون مورداستفاده درزمینه‌ی پیدا کردن کدهای دارای مشکل، فنون یادگیری ماشین[3] و به‌طور خاص‌تر فنون طبقه‌بندی هستند. طبقه‌بندی فرایند تبدیل رکوردهای داده به مجموعه‌ای از کلاس‌ها است. به‌عبارت‌دیگر، طبقه‌بندی شامل پیش‌بینی خروجی بر اساس ورودی است. به هرکدام از این رکوردهای داده، یک نمونه گفته می‌شود. رکوردهای داده از تعدادی معیار یا ویژگی به‌عنوان ورودی و یک برچسب به‌عنوان خروجی تشکیل می‌شوند. هر معیار یا ویژگی یک خصیصه قابل‌اندازه‌گیری از یک پدیده است. برای پیش‌بینی نتایج، ابتدا باید داده‌های موجود واکشی شود. بر اساس این داده‌ها، رکوردها طبقه‌بندی می‌شوند. مجموعه داده به دو مجموعه‌ی آموزش و آزمون تقسیم می‌شود. مجموعه آموزشی شامل داده‌هایی هستند که قبلاً طبقه‌بندی‌شده‌اند و به‌عنوان مرجع برای طبقه‌بندی استفاده می‌شوند. مجموعه آزمون شامل داده‌هایی هستند که قبلاً طبقه‌بندی نشده‌اند و درواقع هدف، طبقه‌بندی آن‌ها است. الگوریتم طبقه‌بندی با تحلیل این داده‌های موجود، نتایج را پیش‌بینی می‌کند. به عبارتی طبقه‌بندی، فرایند یافتن مدلی که توصیف‌کننده کلاس‌ها و مفاهیم داده است و داده‌ها را به گروه‌های مشخص تفکیک می‌کند. الگوریتم‌های طبقه‌بندی، قادر به یادگیری از تجربیات گذشته هستند و این یادگیری بر اساس تجربه نشان‌دهنده یک گام اساسی در تقلید از توانایی‌های استقرایی مغز انسان است که بر اساس این توانایی مغز می‌تواند مسئله شناسایی یک گروه از دسته‌ها یا همان زیرجمعیت‌ها را انجام دهد.

۲-۱-۱. شبکه عصبی

شبکه عصبی[4]، یکی از الگوریتم‌های طبقه‌بندی است که در یادگیری ماشین از آن استفاده می‌شود. شبکه عصبی با ساده‌تر کردن زندگی انسان‌ها در زمینه‌های مختلفی مثل علم پزشکی، اقتصاد، مهندسی و غیره تفاوت‌های زیادی نسبت به شیوه زندگی در چند دهه گذشته ایجاد کرده است. شبکه عصبی، بنای علم یادگیری عمیق[5] و یادگیری عمیق هم به‌نوبه خود، پایه و اساس یادگیری ماشین است. این مفاهیم باهم، علمِ هوش مصنوعی را تشکیل می‌دهند. هدف کلی این است که یک سری اطلاعات طوری به یک ماشین یا همان کامپیوتر داده شود، که برای او قابل‌درک باشد و بتواند از آن در راستای اهداف خواسته‌شده استفاده کند. مفهوم شبکه عصبی مثل این است که بخواهیم به یک کودک یاد بدهیم که چگونه از بین اشکال مختلف، شکل دایره را شناسایی کند. به او چندین عکس از دایره‌ها در ابعاد و رنگ‌های مختلف نشان می‌دهیم. پس از مدتی یاد می‌گیرد که دایره چیست و می‌تواند از میان همه تصاویری که به او نشان داده می‌شود، دایره‌ها را پیش‌بینی و تشخیص دهد. این دقیقاً همان کاری است که به کمک شبکه‌های عصبی برای آموزش به یک ماشین انجام می‌دهیم. آموزش دادن به ماشین درنهایت باعث ایجاد هوش مصنوعی در آن می‌شود. شبکه‌های عصبی داده‌ها را دریافت و در لایه‌های مخفی خود آن‌ها را تحلیل می‌کنند تا درنهایت یک خروجی ارائه دهند. این داده‌ها می‌توانند گروهی از تصاویر، صداها، نوشته‌ها و غیره باشند که باید ترجمه و برای یک ماشین قابل‌درک شوند. به کمک شبکه‌های عصبی، اطلاعات طبقه‌بندی می‌شوند.

۲-۲. انتخاب ویژگی

با پیشرفت سریع فناوری، اکنون مجموعه داده‌هایی با صدها و هزاران متغیر یا ویژگی در پیش‌بینی الگو، داده‌کاوی و یادگیری ماشین وجود دارند. از طرفی، کارایی یک مدل پیش‌بینی کننده، بسیار وابسته به کیفیت مجموعه داده است. در مباحث و مسائل مربوط به پیش‌بینی، یک مجموعه داده معمولاً شامل تعداد زیادی از معیارهای نرم‌افزاری یا همان ویژگی است که ممکن است بسیاری از آن‌ها نامرتبط باشند. ویژگی‌های نامرتبط، اطلاعات مفیدی را برای ساخت مدل ارائه نمی‌دهند. بنابراین،

[1] Long Methods
[2] Feature Envy
[3] Machine Learning
[4] Neural Network
[5] Deep Learning

ممکن است که این دسته از ویژگی‌ها منجر به کاهش کارایی و دقت طبقه‌بندی شوند. شواهد نشان می‌دهد که ویژگی‌ها و معیارهای نرم‌افزاری غیر مرتبط تأثیری بر دقت پیش‌بینی نداشته و حتی در برخی موارد، کارایی را نیز پایین می‌آورد. حذف ویژگی‌ها و معیارهای غیرضروری، می‌تواند با استفاده از الگوریتم‌های انتخاب ویژگی انجام شود. انتخاب ویژگی به معنای فرآیند انتخاب زیرمجموعه‌ای از ویژگی‌های مرتبط، به‌منظور ساخت مدل پیش‌بینی‌کننده است. انتخاب ویژگی، در مدل‌های یادگیری ماشین نقشی مهم ایفا می‌کند و ازآنجایی‌که عموماً مدل پیش‌بینی‌کننده با استفاده از روش‌های یادگیری ماشین ایجاد می‌شود، استفاده از انتخاب ویژگی امری ضروری به نظر می‌رسد. یکی از یافته‌های اصلی در یک بررسی ادبیات موضوع در حوزه پیش‌بینی بوهای کد در نرم‌افزار این است که انتخاب ویژگی، عملکرد مدل‌های پیش‌بینی را که از فنون یادگیری ماشین استفاده می‌کنند بهبود می‌بخشد.

۲-۳. الگوریتم‌های انتخاب ویژگی

الگوریتم‌های انتخاب ویژگی دسته‌بندی‌های متفاوتی دارند. این الگوریتم‌ها برای تولید زیرمجموعه‌های ویژگی می‌توانند از روش‌های جستجوی مختلفی استفاده کنند و بنابراین از این نظر می‌توانند دسته‌بندی‌های مختلفی مانند جستجوی تصادفی و جستجوی کامل داشته باشند. اغلبِ منابع، الگوریتم‌های انتخاب ویژگی را ازنظر معیار ارزیابی هر زیرمجموعه ویژگی، به دودسته‌ی اصلی الگوریتم‌های انتخاب ویژگی مبتنی بر فیلتر[1] و مبتنی بر پوشش[2] تقسیم کرده‌اند. همچنین برخی از منابع، الگوریتم‌های انتخاب ویژگی ترکیبی[3] و الگوریتم‌های انتخاب ویژگی تعبیه‌شده[4] را به‌عنوان دسته‌های دیگر الگوریتم‌های انتخاب ویژگی ارائه کرده‌اند. الگوریتم‌های مورداستفاده در این مقاله عبارت است از چهار الگوریتم Information Gain، ReliefF، FCBF و Information Gain Ratio که جزوه الگوریتم‌های انتخاب ویژگی مبتنی بر فیلتر هستند.

۲-۳-۱. الگوریتم Information Gain

برای چگونگی کارکردی الگوریتم Information Gain، باید مفهوم آنتروپی[5] را خوب دانست. آنتروپی درواقع نشان‌دهنده کم بودن اطلاعات است. یعنی در مجموعه داده، بر اساس یک ویژگی (بُعد) چقدر می‌توان کلاس نهایی را تشخیص داد. درواقع وقتی گفته می‌شود که یک ویژگی دارای آنتروپی بالا است، یعنی آن ویژگی اطلاعات کمتری دارد. الگوریتم Information Gain از آنتروپی هر ویژگی کمک می‌گیرد. درواقع این الگوریتم به میزان اطلاعاتی که می‌توان از یک ویژگی به دست آورد، گفته می‌شود.

فرض بر این است که یک مدیر دانشکده می‌خواهد یک طبقه‌بند بسازد تا با کمک آن طبقه‌بند مشخص کند کدام یک از دانشجوهای دانشکده می‌توانند در آزمون دکتری، قبول شوند(درواقع می‌خواهد یک پیش‌بینی انجام دهد). این پیش‌بینی می‌تواند باعث این شود که از این به بعد دانشجویان با پتاسیلِ بالا را پیدا کرده و روی آن‌ها سرمایه‌گذاری کند. مثلاً به آن‌ها وام‌هایی دهد تا بتوانند مقالات بهتری را در نشریات معتبرتر صادر کنند. درواقع می‌خواهد یک مدل طبقه‌بندی ایجاد کند تا بتواند توسط داده‌های گذشته، یک مدل ساخته و از این به بعد، هر بار که یک دانشجوی جدید به مدل یادگرفته شده داده شد، این مدل بتواند بفهمد که این دانشجو با چه احتمالی می‌تواند در آزمونِ دکتری قبول شود. ویژگی‌هایی که می‌توان برای این مجموعه داده متصور بود شامل معدل کل (عدد)، تعداد مقالات (عدد)، مدرک IELTS زبان‌دارند (۰ یا ۱)، سنوات تحصیلی (عدد) و همچنین یک ویژگی برچسب که اگر دانشجو در مقطع دکتری قبول‌شده بود، بلی و اگر قبول نشده بود، خیر است. همان‌طور که گفته شد در الگوریتم Information Gain هدف این است که فهمیده شود که یک ویژگی خاص چقدر می‌تواند اطلاعات بیشتری دهد. همان‌طور که در مثال قبل دیده شد، ویژگی تعداد مقالات ارائه‌شده (اگر تأثیر بیشتری در دقت مدل داشته باشد) اطلاعات بیشتری نسبت به ویژگی مدرک IELTS می‌دهد. درواقع اگر ویژگی تعداد مقالات به‌عنوان ریشه‌ی درخت انتخاب شود، احتمالاً درخت بهتری وجود دارد و زودتر به نتیجه رسیده می‌شود. پس با این تفاسیر ویژگی تعداد مقالات Gain بیشتری خواهد داشت و امتیاز بیشتری نسبت به سایر ویژگی‌های دیگر کسب می‌کند.

۲-۳-۲. الگوریتم Information Gain ratio

همان‌طور که در الگوریتم قبلی اشاره شد هر چه بهره اطلاعاتی بیشتر باشد به این معنی است که بر اساس آن ویژگی انتخاب‌شده، داده‌ها بهتر تقسیم‌شده‌اند. اما معیار بهره اطلاعاتی به سمت ویژگی‌های با مقادیر بیشتر بایاس دارد که این جزو نقاط ضعف این معیار است. این‌که بهره اطلاعاتی به سمت ویژگی‌های با مقادیر بیشتر بایاس دارد، بر نحوه‌ی کارایی و نتیجه‌ی حاصل از درخت تصمیم تأثیرگذار است. ازاین‌رو، معیار دیگری به نام ضریب بهره ایجاد شده است تا این مشکل را مرتفع سازد. به این صورت که یک ویژگی با چه گستردگی و یکنواختی در مجموعه داده وجود دارد.

[1] Filter-Based Feature Selection
[2] Wrapper-Based Feature Selection
[3] Hybrid Feature Selection
[4] Embedded Feature Selection
[5] Entropy

۲-۳-۳. الگوریتم ReliefF

در این روش، در هر مرحله و به‌طور تصادفی، یک نمونه از میان نمونه‌های موجود در مجموعه داده انتخاب می‌شود. سپس، میزان مرتبط بودن هرکدام از ویژگی‌ها، براساس اختلاف میان نمونه انتخاب‌شده و دو نمونه همسایه نزدیک (کلاس نمونه اول، مشابه کلاس نمونه انتخابی و کلاس نمونه دوم، مخالف کلاس نمونه انتخابی است) به‌روزرسانی می‌شود. اگر یکی از ویژگی‌های نمونه انتخاب‌شده با ویژگی مشابه در نمونه همسایه از کلاس مشابه اختلاف داشته باشد، امتیاز این ویژگی کاهش می‌یابد. از سوی دیگر، اگر همان ویژگی در نمونه انتخاب‌شده با ویژگی مشابه در نمونه همسایه از کلاس مخالف، اختلاف داشته باشد، امتیاز این ویژگی افزایش می‌یابد.

۲-۳-۴. الگوریتم FCBF

روش فیلتر مبتنی بر همبستگی سریع[1]، سرعت و کارایی بهتری نسبت به روش‌های انتخاب ویژگی ReliefF و مبتنی بر همبستگی[2] از خود نشان می‌دهد. در نتیجه، به شکل بهتری می‌تواند خود را با داده‌های با ابعاد بالا منطبق و ویژگی‌های به‌مراتب بهتری را از داده‌های ورودی انتخاب کند. در این روش، ابتدا مقدار عدم قطعیت نامتقارن[3] برای تمامی ویژگی‌ها محاسبه می‌شود. این مقدار، از طریق محاسبه «بهره اطلاعاتی x به‌شرط y» تقسیم‌بر «مجموع کل آنتروپی تمامی ویژگی‌ها» به دست می‌آید. سپس، مقادیر عدم قطعیت نامتقارن، مرتب‌سازی و ویژگی‌های زائد[4] حذف می‌شوند.

۳. کارهای مرتبط

بوچی [۱]، بوی کد را یک ناهنجاری در کد منبع تعریف می‌کند که نقض اصول طراحی اساسی مانند انتزاع، سلسله‌مراتب، کپسوله‌سازی، ماژولاریتی و تغییرپذیری را نشان می‌دهد. در ادامه تعریف این ناهنجاری آمده است که حتی اگر اصول طراحی برای توسعه‌دهندگان شناخته‌شده باشد، اغلب به دلیل بی‌تجربگی، فشار ضرب‌الاجلی و رقابت سنگین در بازار نقض می‌شوند. اوپدیک [۲]، یکی از راه‌های حذف کدهای نابسامان را استفاده از روش‌های بازآرایی کد می‌داند. یعنی فنونی که کد را بدون تغییر در رفتار خارجی نرم‌افزار، بهبود ببخشد. عبدلموز [۳]، نتیجه‌گیری کرده است، به‌منظور پیش‌بینی کدهای نابسامان مانند متد طولانی، زنجیره پیام و پارامتر طولانی در کد نرم‌افزار چندین روش متفاوت وجود دارد که هرکدام از این روش‌ها نتایج متفاوتی را تولید می‌کند.

کسنتینی [۴]، یک راه‌حل موازی مبتنی بر توزیع‌شدگی ارائه می‌دهد. در این راه‌حل چندین روش پیش‌بینی کدهای نابسامان به‌صورت موازی باهم انجام می‌شوند و درنهایت بهینه‌ترین راه‌حل برگزیده می‌شود. بوز و راندل [۵]، دو ابزار پیش‌بینی بوی کد را بر روی پروژه‌ای شامل زنجیره پیام‌ها مقایسه کردند و تفاوت نتایج بین آن‌ها را نشان دادند. با توجه به نتایجی که در تحقیقات بوز و راندل به دست آمد، رسول و آرشاد [۶]، ابزارها و فنون پیش‌بینی موجود را طبقه‌بندی، مقایسه و ارزیابی کردند تا طبقه‌بندی را بهتر درک کنند. با توجه به نتیجه حاصله، سه دلیل اصلی برای نابرابری در نتایج وجود داشت: (۱) توسعه‌دهندگان می‌توانند کدهای نابسامان را به‌صورت ذهنی تفسیر کرده و به روش‌های مختلف شناسایی کنند، (۲) توافق بین آشکارسازها کم است، یعنی چندین ابزار با قوانین مختلف کدهای نابسامان را پیش‌بینی می‌کنند. و (۳) مقدار آستانه برای شناسایی کد نابسامان می‌تواند از یک آشکارساز به آشکارساز دیگر متفاوت باشد.

به‌منظور جمع‌بندی نظرات و ارائه یک طبقه‌بندی که مورد تأیید همه باشد، مارینو [۷]، فنون یادگیری ماشین را برای پیش‌بینی چهار کد نابسامان با کمک ۳۲ فن طبقه‌بندی پیشنهاد کردند. پس از مشاهده نتایج، نویسندگان پیشنهاد کردند که طبقه‌بندی‌کننده‌های یادگیری ماشین مناسب‌ترین روش برای پیش‌بینی کدهای نابسامان هستند. دی نوچی و پالومبا [۸]، به برخی از محدودیت‌های روش مارینو پرداختند. یکی از ایرادهای روش مارینو این بود که منطبق بر دنیای واقعی کدهای برنامه‌نویسی نبود. در برنامه‌نویسی همیشه معیارها و شاخص‌های ثابتی به‌منظور پیش‌بینی کدهای نابسامان وجود ندارد و همیشه این معیارها در حال تغییر است. دیتریچ [۹]، برای اجتناب از این محدودیت و شبیه‌سازی مجموعه داده‌ها، مجموعه داده‌های دو روش قبل را دوباره باهم ترکیب کردند و مجموعه داده‌های سطح کلاس و سطح متد را باهم ادغام و دوباره پیکره‌بندی کردند. با این ادغام توانستند بیش از یک نوع کد نابسامان را تشخیص دهند. همچنین همان فنون یادگیری ماشینِ روش مارینو را بر روی مجموعه داده‌های اصلاح‌شده آزمایش کردند و به‌طور متوسط ٪۷۶ دقت در همه مدل‌ها به دست آمد.

عظیم و پالومبا [۱۰]، نتیجه‌گیری کردند که بنابر تجزیه‌وتحلیل‌های انجام‌شده، کلاس خدا، متد طولانی، تجزیه عملکردی و کد اسپاگتی به‌شدت در ادبیات موضوع موردتوجه قرارگرفته‌اند. همچنین بیان کردند که درختان تصمیم و ماشین‌های بردار پشتیبان رایج‌ترین الگوریتم‌های یادگیری ماشین برای تشخیص کدهای نابسامان هستند. JRip و جنگل تصادفی مؤثرترین طبقه‌بندی‌کننده‌ها ازنظر عملکرد هستند. همچنین تجزیه‌وتحلیل‌های مختلف، وجود چندین موضوع و چالش باز را نشان می‌دهد که جامعه پژوهشی باید در آینده بر

[1] Fast Correlation-based Filter
[2] Correlation-based
[3] Symmetrical Uncertainty
[4] Redundant Features

آن‌ها تمرکز کند.

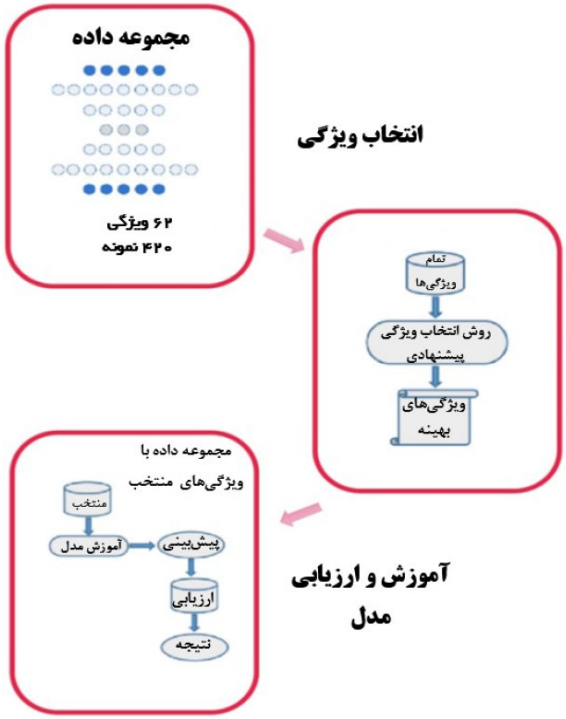
کریمر [۱۱]، یک روش تطبیقی به‌منظور یافتن عیوب طراحی (کلاس بزرگ و متد طولانی) با ترکیب روش‌های شناخته‌شده بر اساس معیارهای نرم‌افزار با استفاده از فنون طبقه‌بندی به نام درخت تصمیم معرفی کرد. خومه و ووچر [۱۳]، یک روش بیزی برای پیش‌بینی کدهای نابسامان در نرم‌افزارهای منبع باز پیشنهاد کردند. آن‌ها همچنین روش معیار پرسش هدف را برای ساخت شبکه‌های بیزی از تعاریف ضد الگوها و اعتبارسنجی با ضد الگوها در دو برنامه منبع باز ارائه کردند.

باچاریا و مایگا [۱۴]، روش پیش‌بینی ماشین بردار پشتیبان را برای پیش‌بینی ضد الگوها معرفی کردند. اموریم و آنتونس [۱۵]، شناسایی کدهای نابسامان را از طریق الگوریتم‌های درخت تصمیم موردمطالعه قراردادند. والتر [۱۶]، فنون طبقه‌بندی یادگیری ماشین را روی چهار مجموعه داده کد نابسامان (یعنی کلاس داده، متد طولانی، حسادت ویژگی، کلاس خدا) آزمایش کردند تا آن‌ها را شناسایی کنند. برای این کار، نویسندگان از ۷۴ پروژه توسعه داده‌شده با زبان جاوا استفاده کرده‌اند که متعلق به Qualitus Corpus است.

زنونی [۱۷] با استفاده از روش یادگیری ماشینی، کدهای نابسامان را طبقه‌بندی کرد. با استفاده از این روش، توانست کدهای نابسامان را در سطح کلاس و متد دسته‌بندی کند. دی نوچی و پالومبا [۱۸]، برخی از ایرادهای فونتانا و همکاران را تصحیح کردند. آن‌ها در یافته‌های خود به این نتیجه رسیدند که این مشکل به دلیل استفاده از یک مجموعه داده خاص به‌جای قابلیت‌های واقعی الگوریتم‌های یادگیری ماشین است. به این صورت که مجموعه دادگان فونتانا دوبار پیکربندی گردید و مجموعه داده‌های جدیدی ارائه شد. به این نتیجه رسیدند که هنوز با الگوریتم‌های یادگیری ماشین نمی‌توان تمام کدهای نابسامان را تشخیص داد و هنوز این حوزه نیاز به توجه و کار بیشتر دارد. پکرولی و دی‌نوچی [۱۹]، چندین فنون برای رسیدگی به مسائل عدم تعادل داده برای درک تأثیر آن‌ها بر طبقه‌بندی‌کننده‌های یادگیری ماشین برای پیش‌بینی کدهای نابسامان را بررسی کردند.

۴. روش پیشنهادی

فرآیند روش پیشنهادی در شکل (۱) نشان داده شده است. همان‌طور که مشاهده می‌شود ابتدا برای بهینه کردن مجموعه داده، عمل انتخاب ویژگی روی آن انجام شده و درنهایت مدل شناسایی کدهای نابسامان روی مجموعه داده‌ی بهینه‌شده ساخته می‌شود.



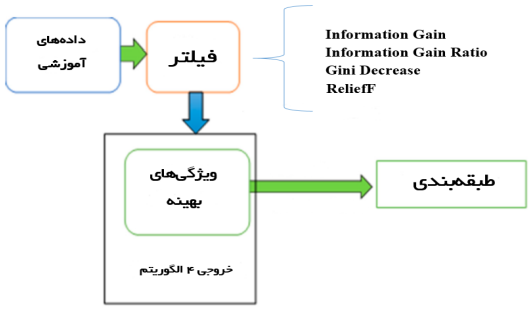
شکل (۱). فرایند کلی روش پیشنهادی

مطابق شکل (۱)، فرایند جمع‌آوری داده از منابع موثق که منتج به تشکیل یک مجموعه داده قابل‌اتکا می‌شود، انجام می‌گیرد. انتخاب ویژگی با حذف ویژگی‌های غیر مرتبط و تکراری به کاهش ابعاد داده کمک می‌کند. در این مرحله با استفاده از روش‌های مرسوم در انتخاب ویژگی، ویژگی‌هایی که بیشترین کمک را به مدل پیش‌بینی می‌کنند، انتخاب می‌شوند. یک انتخاب ویژگی صحیح می‌تواند منجر به بهبود یادگیرنده از جهات گوناگون از جمله سرعت یادگیری، ظرفیت تعمیم و سادگی مدل استنتاج شود. انتخاب ویژگی برای کاربردهایی که در آن‌ها ویژگی‌های اصلی برای تفسیر مدل و استخراج دانش مهم هستند بسیار کاربردپذیر است. زیرا طی این فرآیند، ویژگی‌های اصلی مجموعه داده حفظ می‌شوند. در ادامه مبانی انتخاب ویژگی شرح داده خواهد شد.

۴-۱. ترکیب الگوریتم‌های انتخاب ویژگی

مطابق شکل (۲)، هدف اصلی در این مرحله عبارت است از انتخاب زیرمجموعه‌ای بهینه از ویژگی‌ها به‌طوری‌که دقت مدل پیش‌بینی کدهای نابسامان را افزایش دهد. وقتی در مورد انتخاب ویژگی صحبت می‌شود الگوریتمی که برای این انتخاب ویژگی مورداستفاده قرارگرفته است ازنظر نوع انتخاب هر ویژگی مشخصات کارکردی مختص به خود را دارد. یعنی هر الگوریتم با

توجه به قوانینی که برای انتخاب ویژگی دارد، بهترین ویژگی را که سبب افزایش دقت مدل می‌شود، انتخاب می‌کند. نمی‌توان گفت که الگوریتمی خاصی در حوزه انتخاب ویژگی می‌تواند بهترین ویژگی‌ها را از مجموعه داده استخراج کند. ایده‌ای که در این زمینه می‌توان بررسی کرد این است که از منظر چند الگوریتم انتخاب ویژگی، به این مسئله نگاه کرد و در انتها خروجی این الگوریتم‌ها را باهم جمع‌بندی کرده و بهترین ویژگی‌هایی را که مورد تأیید الگوریتم‌های کاندید انتخاب ویژگی است باهم به‌منظور یادگیری مدل مورداستفاده قرارداد. بدین منظور، از یک روش با ترکیب چهار الگوریتم انتخاب ویژگی مبتنی بر فیلتر استفاده می‌شود. به این صورت که در مرتبه اول، بعد از رتبه‌دهی به ویژگی‌ها، ویژگی‌های منتخب توسط چهار الگوریتم Information Gain، Information Gain Ratio، Gini Decrease و ReliefF گردآوری می‌شود. در مرحله بعد آستانه‌ای برای انتخاب ویژگی‌های کاندید انتخاب می‌شود. در هر یک از چهار الگوریتم، بعد از انتخاب ویژگی‌ها و پس از بررسی امتیازهای داده‌شده به ویژگی‌ها، آستانه‌ای برای انتخاب ویژگی‌ها در نظر گرفته می‌شود که در این مقاله عدد ۰/۰۰۴ لحاظ گردید. دلیل انتخاب این عدد آن است که بعد از ۰/۰۰۴ تقریباً امتیازها صفر می‌شوند و تأثیری در دقت مدل و فرایند شناسایی نخواهند داشت.



شکل (۲). فرآیند انتخاب ویژگی

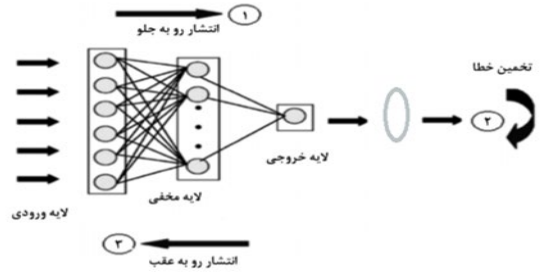
در این مرحله از هرکدام از الگوریتم‌ها چند ویژگی انتخاب می‌شود که امکان دارد باهم اشتراکاتی داشته باشند. به‌منظور افزایش دقت مدل، اجتماع تمامی ویژگی‌هایی که توسط چهار الگوریتم انتخاب شده است به‌عنوان مجموعه ویژگی‌های منتخب وارد مرحله بعد که مرحله طبقه‌بندی است، می‌شود.

۴-۲. استفاده از طبقه‌بند پیشنهادی

همان‌طور که گفته شد طبقه‌بندی علمی است که براساس داده‌های قبلی که دارای برچسب هستند، مدلی برای پیش‌بینی برچسب داده‌های جدید می‌سازد. به‌بیان‌دیگر، طبقه‌بندی فرایند قراردادن نمونه‌های جدید در طبقات مختلف بر اساس داده‌های قدیمی است و برای این کار به یک مدل طبقه‌بند یا الگوریتم طبقه‌بند نیاز است. طبقه‌بندی که برای این کار انتخاب شده است شبکه عصبی پرسپترون چندلایه پس انتشار خطا است.

یک شبکه عصبی پرسپترون چندلایه از پشت هم قرار دادن چند پرسپترون حاصل خواهد شد. یعنی در چنین شبکه‌ای چندلایه از نورون‌ها را خواهیم داشت. الگوریتم پس انتشار خطا هر دو مسیر forward و backward در شبکه حرکت می‌کند و می‌تواند مقدار گرادیانت خطا را نسبت به هر پارامتر شبکه(هر وزن یا بایاس محاسبه کند). به‌این‌ترتیب الگوریتم پس انتشار خطا می‌تواند تعیین کند که مقدار هر وزن در یک شبکه عصبی mlp چقدر باید تغییر کند. و به این ترتیب مسئله آموزش وزن‌های میانی در mlp چندلایه حل می‌شود.

عملکرد الگوریتم backpropagation به‌صورت خلاصه در ادامه آورده شده است. این الگوریتم ابتدا در مسیر forward یک پیش‌بینی انجام می‌دهد و خطا را محاسبه می‌کند. سپس مقدار خطا در جهت backward حرکت می‌کند. برای هر اتصال یک گرادیانِ خطا محاسبه می‌شود. سپس مقدار هر وزن در جهت کاهش خطا تغییر می‌کنند.شکل(۳) شبکه عصبی پرسپترون چندلایه پس انتشار خطا را نشان می‌دهد.



شکل (۳). شبکه عصبی پرسپترون چندلایه پس انتشار خطا

نکته‌ای که بسیار حائز اهمیت است این است که، در شبکه-های عصبی نرون‌های لایه مخفی با تعداد ویژگی‌های لایه ورودی باید برابر باشد. دلیل این امر آن است که اگر تعداد نرون لایه مخفی کمتری انتخاب شود، منجر به کم برازشی[1] می‌شود. در حالی که اگر تعداد نرون‌های زیادی انتخاب شود ممکن است منجر به بیش‌برازشی[2]، واریانس بالا و افزایش زمان لازم برای آموزش شبکه شود. پس بنا بر آنچه گفته شد تعداد نرون‌ها به تعداد ویژگی‌های منتخب است.

شبکه‌ی عصبی به دنبالِ یادگیری از طریقِ تغییرات در وزن‌ها (W) و انحراف‌ها (b) است. اصل و پایه‌ی یادگیری در شبکه‌های عصبی با تکرار[3] انجام می‌شود. یعنی چندین مرتبه داده‌های یک مجموعه داده به الگوریتم تزریق می‌شود و این الگوریتم با کم و زیاد کردن وزن‌ها و انحراف‌ها، می‌تواند تفاوت‌ها را در داده‌های

[1] Underfitting
[2] Overfitting
[3] Iteration

آموزشی تشخیص دهد .یکی از روش‌های بسیار پرکاربرد برای تکرار در شبکه‌های عصبی روشِ پس انتشارِ خطا[1] است. در این روش، در هر دور (یعنی در هر تکرار) دو مرحله خواهیم داشت. مرحله‌ی اول حرکت رو به جلو[2] است که با ضرب داده‌های ورودی در وزن‌ها و سپس جمع آن با انحراف انجام می‌شود. سرانجام در همان مرحله‌ی اول به یک خروجی می‌رسیم که احتمالاً با خروجی واقعی تفاوت دارد. اینجا است که توسط تابع ضرر مشخص می‌کنیم که مرحله‌ی حرکت رو به جلو چه مقدار خطا داشته است .حال که مشخص شد الگوریتم با توجه به وزن‌ها و انحراف‌ها چه مقدار خطا دارد، به مرحله‌ی دوم در یک تکرار می‌رویم. در این مرحله می‌توانیم به عقب بازگشته و وزن‌ها و انحراف‌ها را به هنگام سازی کنیم. یعنی وزن‌ها و انحراف‌ها را به شکلی تغییر دهیم تا در تکرارِ بعدی نتیجه‌ای نزدیک‌تر به خروجیِ واقعی و با خطای کم‌تر را تولید کند. این تکرار حرکت روبه‌جلو و رو به عقب[3] آن‌قدر انجام می‌شود تا خروجی شبکه برای تمامی داده‌های آموزشی، به نزدیک‌ترین مقدارِ واقعیِ خود (یعنی مقداری که توسط داده‌های آموزشی در اختیار داریم) برسد. به‌این‌ترتیب الگوریتم یاد گرفته است و می‌تواند با مشاهده‌ی ویژگی‌های یک داده جدید مشخص کند متعلق به کدام دسته است.

تعداد تکراری که در این مقاله روی این شبکه عصبی اعمال شده است ۸۰ تکرار است. اهمیت وجود توابع فعال‌ساز به این خاطر است که تابع فعال‌ساز تصمیم می‌گیرد که هر نرون، فعال شود یا خیر. اگر از توابع فعال‌ساز استفاده نشود، وزن‌ها و مقدار بایاس فقط یک معادله‌ی خطی را ایجاد می‌کنند. درست است که معادله‌ی خطی خیلی راحت‌تر قابل‌حل است، اما برای حل مسائل پیچیده کمکی نمی‌کند؛ درواقع معادلات خطی در یادگیری الگوهای پیچیده‌ی داده خیلی محدود هستند و یک شبکه‌ی عصبی بدون تابع فعال‌ساز فقط یک مدل رگرسیون خطی است. به‌طورکلی، شبکه‌های عصبی از توابع فعال‌ساز استفاده می‌کنند تا بتوانند به شبکه در یادگیری داده‌های پیچیده کمک و پیش‌بینی قابل‌قبولی را در خروجی ارائه کنند. تابع فعال‌سازی که در این شبکه عصبی استفاده شده است تابع فعال‌ساز ReLU است که از توابع فعال‌ساز غیرخطی است.

۵. ارزیابی روش پیشنهادی

در این مقاله، یک روش جدید انتخاب ویژگی با ترکیب چند الگوریتم انتخاب ویژگی مبتنی بر فیلتر و اجتماع نتایج آن‌ها و استفاده از شبکه عصبی به‌عنوان طبقه‌بند ارائه گردید. برای ارزیابی این روش، از اعتبارسنجی متقابل[4] استفاده شده و نتیجه آن با حالت بدون انتخاب ویژگی و به همراه انتخاب ویژگی الگوریتم‌های موجود مقایسه شده است. مجموعه داده‌ای که برای این تحقیق استفاده شده است در ادامه توضیح داده خواهد شد.

۵-۱. مجموعه داده‌های مورد استفاده

در این مقاله، دو مجموعه داده برای هر دو کد نابسامان شامل متد طولانی و خصیصه حسادت از فونتانا استفاده شده است.

برای ساخت این مجموعه داده از ۱۱۱ پروژه نرم‌افزاری استفاده شده است. از این تعداد ۳۷ پروژه به دلایلی از جمله عدم کامپایل شدن، حذف شده و تعداد ۷۴ پروژه برای استفاده در تشخیص کدهای نابسامان استفاده شده‌اند. مشخصات کامل این ۷۴ پروژه در جدول (۱) آمده است.

۵-۲. معیارهای ارزیابی

تعداد نمونه‌هایی که در این مجموعه داده قرار دارد ۴۲۰ نمونه است. تعداد سنجه‌ها و شاخص‌های موجود هم ۶۲ شاخص است.

جدول (۱). مشخصات مجموعه داده

ردیف	تعداد پروژه	تعداد کل خطوط کد	تعداد کل بسته‌ها	تعداد کلاس‌ها	تعداد کل متدها
۱	۷۴	۶,۷۸۵,۵۶۸	۳۴۲۰	۵۱,۸۲۶	۴۰۴,۳۱۶

در جدول (۲) برخی از این شاخص‌ها توضیح داده شده است.

جدول(۲). تعریف کدهای نابسامان

Lines of Code (LOC)	
تعریف	تعداد خطوط کد یک عملیات یا یک کلاس، ازجمله خطوط خالی و توضیحات
محاسبه و تقسیم‌بندی	**متد**: از ابتدای شروع براکت تا انتهای براکت **متد کلاس**: از ابتدای تعریف یک کلاس تا انتهای تعریف کلاس **بسته**: تعداد خطوط کد مربوط به کلاس‌های تعریف‌شده در یک بسته **کلان پروژه** : مجموع تمام خطوط کد تمام بسته‌ها
Lines of Codes Without Accessor or Mutator Methods (LOCNAMM)	
تعریف	تعداد خطوط کد یک کلاس، شامل خطوط خالی و توضیحات، به‌استثنای متدهای accessor و mutator و توضیحات مربوطه.
محاسبه و تقسیم‌بندی	**کلاس**: از ابتدای تعریف یک کلاس تا انتهای تعریف کلاس ایجاد می‌گردد)IDE معمولاً توسط Setter و Getter(
Number of Packages (NOPK)	
تعریف	تعداد بسته‌های موجود در کل پروژه
محاسبه و تقسیم‌بندی	ما یک وابستگی به کلاس‌های اعلام‌شده در کل پروژه را اعلام می‌کنیم. به‌این‌ترتیب تمام بسته‌هایی که ذخیره DFMC4Jشامل کلاس‌ها هستند را در مدل می‌کنیم.

[1] Back propagation of error
[2] Feed forward
[3] Back propagation
[4] Cross Validation

۶. نتیجه‌گیری

نتایج ارزیابی نشان می‌دهد که روش پیشنهادی قادر است با انتخاب ویژگی بهینه با استفاده از طبقه‌بند موردنظر (شبکه عصبی) معیارهای ارزیابی خواسته‌شده و مدنظر را تحقق بخشیده و آن‌ها را در مورد بوی کد متد طولانی بهبود دهد.

در جدول (۳) نتایج و مقایسه طبقه‌بندی از طریق انتخاب ویژگی با استفاده از هرکدام از چهار روش انتخاب ویژگی به‌صورت مجزا نشان داده شده است. در انتهای جدول روش پیشنهادی آمده است که نتیجه ادغام چهار روش انتخاب ویژگی است.

جدول (۳). مقایسه نتایج طبقه‌بندی با استفاده از چهار روش انتخاب ویژگی و روش پیشنهادی

الگوریتم	AUC	CA	F1	Precision	Recall
ReliefF	٪۸۲,۴	٪۷۸,۰	٪۷۵,۸	٪۸۰,۲	٪۷۸,۰
IG	٪۹۵,۹	٪۸۵,۶	٪۸۴,۷	٪۸۷,۸	٪۸۵,۶
IGR	٪۹۱,۶	٪۸۴,۹	٪۸۴,۶	٪۸۵,۰	٪۸۴,۹
FCBF	٪۹۸,۰	٪۸۷,۲	٪۸۶,۵	٪۸۸,۷	٪۸۷,۲
روش پیشنهادی	٪۹۹,۶	٪۹۸,۶	٪۹۸,۵	٪۹۸	٪۹۹,۷

در جدول (۴)، نتایج طبقه‌بندهای مختلف روی همین مجموعه داده و به‌منظور پیش‌بینی و شناسایی کد نابسامان متد طولانی آورده شده است.

جدول (۴). نتایج طبقه‌بندهای مختلف روی همین مجموعه داده

طبقه‌بند	Accuracy	F-measure	ROC area
B-RandomForest	٪۹۵,۹	٪۹۶,۰	٪۹۷,۶
RandomForest	٪۹۵,۹	٪۹۶,۰	٪.۹۷
B-J48UnPruned	٪۹۵,۴	٪۹۵,۵	٪۹۷,۱
B-J48Pruned	٪۹۴,۷	٪۹۴,۸	۹۷,۷
J48Unpruned	٪۹۳,۵	٪۹۳,۵	٪۹۱,۹

به‌منظور ارزیابی دقت طبقه‌بندی از روش اعتبارسنجی متقابل استفاده شده است. اگر مجموعه داده‌های آموزشی را به‌طور تصادفی به k زیر نمونه یا دسته [1]با حجم یکسان تفکیک کنیم، می‌توان در هر مرحله از فرایند اعتبارسنجی متقابل، تعداد k-1 از این دسته‌ها را به‌عنوان مجموعه داده آموزشی و یکی را به‌عنوان مجموعه داده اعتبارسنجی (آزمون) در نظر گرفت. تصویر زیر، مراحل روش k-Fold را به‌خوبی نشان می‌دهد. مشخص است که با انتخاب k=10، تعداد تکرارهای فرآیند اعتبارسنجی برابر با ۱۰ خواهد بود و دستیابی به مدل مناسب امکان‌پذیر می‌شود.

جدول (۵).نتایج طبقه‌بندی با استفاده از روش اعتبارسنجی متقابل

تکرار	Recall	Precision	F1	CA	AUC
۱	۹۷.٪۸	۹۷.٪۸	۹۷.٪۸	۹۷.٪۸	۹۹.٪۸
۲	۹۹.٪۳	۹۷.٪۳	۹۹.٪۳	۹۹.٪۳	۹۹.٪۸
۳	۹۹.٪۰	۹۹.٪۰	۹۹.٪۰	۹۹.٪۰	۹۹.٪۰
۴	۹۸.٪۸	۹۸.٪۷	۹۸.٪۷	۹۸.٪۸	۹۹.٪۷
۵	۹۸.٪۸	۹۸.٪۸	۹۸.٪۸	۹۸.٪۸	۹۹.٪۷
۶	۹۹.٪۰	۹۷.٪۰	۹۹.٪۰	۹۹.٪۰	۹۹.٪۷
۷	۹۸.٪۵	۹۷.٪۵	۹۸.٪۵	۹۸.٪۵	۹۹.٪۶
۸	۹۸.٪۳	۹۸.٪۳	۹۸.٪۳	۹۸.٪۳	۹۹.٪۹
۹	۹۸.٪۳	۹۸.٪۳	۹۸.٪۳	۹۸.٪۳	۹۹.٪۹
۱۰	۹۷.٪۸	۹۷.٪۸	۹۷.٪۸	۹۹.٪۰	۹۹.٪۶
میانگین	۹۹.٪۷	٪۹۸	۹۸.٪۵	۹۸.٪۶	۹۹.٪۶

نتیجه‌گیری نهایی این است که پیش‌بینی زودهنگام کدهای نابسامان در کد منبع برنامه، منجر به بهبود کیفیت نرم‌افزار می‌شود. امکان تحقق این هدف، با استفاده از روش پیشنهادی و به‌تبع آن، ارتقاء قابلیت درک، تکامل و نگهداشت‌پذیری برنامه و جلوگیری از شکست احتمالی در آینده، وجود دارد

روش‌های انتخاب ویژگی مناسب تأثیر بسزایی در کیفیت و دقت پیش‌بینی مدل طبقه‌بند دارند. اگر ویژگی‌ها به‌صورت بهینه انتخاب شوند، هم در زمان و هم در دقت مدل شاهد رشد چشم‌گیری خواهیم بود. یکی از مشکلاتی که عملکرد مدل‌های پیش‌بینی کدهای نابسامان در نرم‌افزار را تحت تأثیر قرار می‌دهد، ابعاد بالای داده است. ابعاد بالای داده به معنی وجود ویژگی‌های افزونه و غیرمفید است که باعث کاهش دقت مدل‌های پیش‌بینی کدهای نابسامان می‌شود. برای حل این مشکل محققان از الگوریتم‌های انتخاب ویژگی استفاده می‌کنند. در این مقاله نشان دادیم که انتخاب ویژگی بر اساس روش پیشنهادی از طریق اجماع نظر چهار الگوریتم یادشده در بخش مرور ادبیات تحقیق با تشکیل مجموعه ناظرهایی، توانستیم به خروجی مدنظر که همان ویژگی‌های بهینه است، دست یابیم. طبقه‌بند استفاده‌شده برای دسته‌بندی هم بسیار حائز اهمیت است که در این مقاله از پرسپترون چندلایه با انتشار رو به عقب خطا برای طبقه‌بندی کدهای نابسامان استفاده گردید.

[1] Fold

همچنین، از الگوریتم‌های یادگیری ماشین به‌منظور پیش‌بینی کدهای نابسامان استفاده کردیم. اما با توجه به اینکه روزبه‌روز الگوریتم‌های بیشتری از مدل‌های یادگیری ماشین ظهور می‌کنند، می‌توان موارد زیر را مورد بررسی قرار داد:

۱- آیا فنون دیگر می‌توانند پیش‌بینی بهتری از کدهای نابسامان انجام دهند یا خیر. از سوی دیگر، مجموعه داده‌ای که برای ارزیابی این طرح استفاده شده است کوچک است. تولید مجموعه داده بزرگ‌تر یا پیدا کردن مجموعه داده‌ای که بتوان روی آن مدل را آموزش داد می‌تواند به غنی‌تر کردن کار کمک نماید.

۲- این مقاله فقط می‌تواند دو نوع کد نابسامان را پوشش دهد. روش‌های دیگر هم می‌توانند چهار نوع کد نابسامان را تشخیص دهند. روشی که بتواند تعداد بیشتری از کدهای نابسامان را به‌صورت هم‌زمان پوشش دهد، انجام نشده است.

۳- کدهای نابسامانی هستند که هنوز با الگوریتم‌های یادگیری ماشین نمی‌توان آن‌ها را پیش‌بینی کرد. بنابراین، در آینده می‌توان، روش‌هایی را ارائه کرد که قادر باشند این‌گونه کدها را شناسایی کنند.

ساختاربندی مقاله به این صورت است که در ابتدای مقاله به تعریف مفهوم طبقه‌بندی و موضوعات مرتبط با آن از قبیل شبکه عصبی، انتخاب ویژگی و الگوریتم‌های انتخاب ویژگی پرداخته شد. سپس در بخش بعدی به بررسی کارهای مرتبط در حوزه طبقه‌بندی و الگوریتم‌های استفاده شده به‌منظور انتخاب ویژگی در حل مسائل مرتبط با طبقه‌بندی پرداخته شد و چالشی که در مورد مسئله مربوط به انتخاب الگوریتم انتخاب ویژگی وجود دارد، مطرح گردید. در ادامه روش پیشنهادی به‌منظور حل این مسئله موردبررسی قرار گرفت. و در انتها این روش پیشنهادی مورد ارزیابی قرار گرفت.

۷. مراجع

[1] Booch, G. (2018). Object-oriented analysis and design. Addison-Wesley.Boutell, M.R., Luo, J., Shen, X., Brown, C.M. DOI: [10.1234/abcd](https://doi.org/10.1234/abcd)

[2] Opdyke, W.F. (2022). Refactoring: a program. restructuring aid in designing object-oriented application frameworks PhD thesis. PhD thesis: University of Illinois at Urbana-Champaign .DOI: [10.5678/efgh](https://doi.org/10.5678/efgh)

[3] Abdelmoez, W., Kosba, E., Iesa, A.F. (2016). Risk-based code smells detection tool. In The international Conference on computing technology and information management (ICCTIM2014) (pp. 148–159): The Society of Digital Information and Wireless Communication .DOI: [10.9876/ijkl](https://doi.org/10.9876/ijkl)

[4] Kessentini, W.; Kessentini, M.; Sahraoui, H.; Bechikh, S.; Ouni, A.: A cooperative parallel search-based software engineering approach for code-smells detection. IEEE Trans. Softw. Eng.40(9), 841–861 (2014) DOI: [10.5432/mnop](https://doi.org/10.5432/mnop)

[5] Bowes, D., Randall, D., Hall, T. (2013). The inconsistent measurement of message chains. In 2013 4th International workshop on emerging trends in software metrics (WETSoM) (pp. 62–68): IEEE .DOI: [10.7654/qrst](https://doi.org/10.7654/qrst)

[6] Rasool, G., & Arshad, Z. (2015). A review of code smell mining techniques. Journal of Software: Evolution and Process, 27(11), 867–895 .DOI: [10.2319/uvwx](https://doi.org/10.2319/uvwx)

[7] Fontana, F.A., M¨antyl¨a, M.V., Zanoni, M., Marino, A. (2016b). Comparing and experimenting machine learning techniques for code smell detection. Empirical Software Engineering, 21(3), 1143–1191 .DOI: [10.5555/yzab](https://doi.org/10.5555/yzab)

[8] Di Nucci, D., Palomba, F., Tamburri, D.A., Serebrenik, A., De Lucia, A. (2018). Detecting code smells using machine learning techniques: are we there yet? In 2018 IEEE 25th International conference on software analysis, evolution and reengineering SANER (pp. 612–621):IEEE .DOI: [10.9876/cdef](https://doi.org/10.9876/cdef)

[9] Fontana, F.A., Dietrich, J., Walter, B., Yamashita, A., Zanoni, M. (2016a). Antipattern and code smell false Positives: preliminary conceptualization and classification. In 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER), (Vol. 1 pp.609–613): IEEE .DOI: [10.5432/ghij](https://doi.org/10.5432/ghij)

[10] Azeem, M.I., Palomba, F., Shi, L., Wang, Q. (2019). Machine learning techniques for code smell detection: a systematic literature review and meta-analysis. Information and Software Technology. DOI: [10.8765/klmn](https://doi.org/10.8765/klmn)

[11] Kreimer, J. (2005). Adaptive detection of design flaws. Electronic Notes in Theoretical Computer Science,141(4), 117–136. DOI: [10.9876/pqrs](https://doi.org/10.9876/pqrs)

[12] Khomh, F., Vaucher, S., Gueh ´ eneuc, Y.G., Sahraoui, H. (2009). A Bayesian approach for the detection of ´code and design smells. In 9th International conference on quality software,2009.QSIC'09(pp.305–314): IEEE DOI: [10.9876/tuvw](https://doi.org/10.9876/tuvw)

[13] Khomh, F., Vaucher, S., Gueh ´ eneuc, Y.G., Sahraoui, H. (2011). Bdtex: a gqm-based Bayesian approach for ´the detection of antipatterns. Journal of Systems and Software, 84(4), 559–572. DOI: [10.9876/xyzab](https://doi.org/10.9876/xyzab)

[14] Maiga, A., Ali, N., Bhattacharya, N., Sabane, A., Gu ´ eh ´ eneuc, Y.G., Antoniol, G., A ´ ¨ımeur, E. (2012). Support vector machines for anti-pattern detection. In 2012 Proceedings of the 27th IEEE/ACM international conference on automated software engineering (ASE) (pp. 278281) DOI: [10.9876/defgh](https://doi.org/10.9876/defgh)

[15] Amorim, L., Costa, E., Antunes, N., Fonseca, B., Ribeiro, M. (2015). Experience report: evaluating the effectiveness of decision trees for detecting code smells. In 2015 IEEE 26th international symposium on software reliability engineering (ISSRE) (pp. 261–269): IEEE DOI: [10.9876/ijklm](https://doi.org/10.9876/ijklm)

[16] Fontana, F.A., Dietrich, J., Walter, B., Yamashita, A., Zanoni, M. (2016a). Antipattern and code smell false positives: preliminary conceptualization and classification. In 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER), (Vol. 1 pp. 609–613): IEEE. DOI: [10.9876/mnopq](https://doi.org/10.9876/mnopq)

[17] Fontana, F.A., & Zanoni, M. (2017). Code smell severity classification using machine learning techniques. Knowledge-Based Systems, 128, 43–58. DOI: [10.9876/qrstuv](https://doi.org/10.9876/qrstuv)

[18] Di Nucci, D., Palomba, F., Tamburri, D.A., Serebrenik, A., De Lucia, A. (2018). Detecting code smells using machine learning techniques: are we there yet? In 2018 IEEE 25th International conference on software analysis, evolution and reengineering SANER (pp. 612–621): IEEE DOI: [10.9876/wxyz](https://doi.org/10.9876/wxyz)

[19] Pecorelli, F., Di Nucci, D., De Roover, C., De Lucia, A. (2019a). On the role of data balancing for machine learning-based code smell detection. In Proceedings of the 3rd ACM SIGSOFT international workshop on machine learning techniques for software quality evaluation (pp. 19–24): ACM. DOI: [10.9876/abcdef](https://doi.org/10.9876/abcdef)