Scientific Journal of Electronic & Cyber Defense
Vol. 12, No. 1, Spring 2024, Serial No. 45
ISSN: 2322-4347 , E-ISSN: 2980-8979

Improving the accuracy of code smell identification using the gray wolf algorithm based on machine learning techniques and majority voting

A.Karimi , M.R.Khosravi Farsani,

* Assistant professor , Imam Hossein University, Tehran, Iran

(Received: 2023/03/03, Revised: 2024/04/11, Accepted: 2024/05/11, Published: 2024/06/02)

DOR: https://dorl.net/dor/20.1001.1.23224347.1403.12.1.9.7

ABSTRACT

A code smell refers to the problems and weaknesses in the software. Sometimes code smells are caused by poor design and programming style. The presence of code smell may slow down processing, reduce code quality, increase the risk of program crashes and software errors. Code smell detection is a major challenge in the field of programming and it shows that code smell detection is a complex problem and its identification and solution requires suitable and adaptable methods and algorithms. Detecting code smell improves the quality and provides the ability to develop and modify the code and prevents the occurrence of problems and errors in the future. To identify and fix code smells, researchers have used feature selection and machine learning algorithms. In this article, the gray wolf meta-heuristic algorithm is used to select useful and non-additive features from among the many features of the software. Choosing relevant and appropriate features makes the model simpler, improves accuracy and reduces its training time. To identify and classify code smells, a combination of three nearest neighbor algorithms, decision tree and support vector machine have been used. The final result of the classification is determined based on the majority voting mechanism. The use of the combination of three machine learning algorithms and the use of the majority voting mechanism is the innovation of this article. In order to evaluate the proposed method, the Fontana data set, which includes long method, feature Envy, God class and data class code smells, has been used. The evaluation results of the proposed method are 95, 93.5, 93 and 98.5% based on the accuracy criteria, respectively, which show better performance than the stand-alone support vector machine methods with different kernels, nearest neighbor, decision tree, etc.

Keywords: Code Smell, Feature Selection, Gray Wolf Algorithm, Classification, Machine Learning, Majority Voting.

*Corresponding Author Email: mkarticles90@gmail.com

نشریه علمی "پدافند الکترونیکی و سایبری"

سال دوازدهم، شماره ۱، بهار ۱۴۰۳، ص۱۰۹-۱۲۲

شاپا الکترونیکی: ۸۹۷۹-۲۹۸۰ شاپا چاپی: ۴۳۴۷-۲۳۲۲

علمی– پژوهشی

بهبود دقت شناسایی بوی کد با استفاده از الگوریتم گرگ خاکستری مبتنی‌بر فنون یادگیری ماشین و رای‌گیری اکثریت

علی کریمی[۱]، محمدرضا خسروی فارسانی[۲]

۱- استادیار ۲-کارشناسی ارشد دانشگاه جامع امام حسین، تهران، ایران

(دریافت: ۱۴۰۲/۱۲/۱۳، بازنگری: ۱۴۰۳/۰۱/۲۳، پذیرش: ۱۴۰۳/۰۲/۲۲، انتشار: ۱۴۰۳/۰۳/۱۳)

DOR: https://dorl.net/dor/20.1001.1.23224347.1403.12.1.9.7

ناشر: دانشگاه جامع امام حسین (ع) نویسندگان

چکیده

بوی کد به مشکلات و ضعف‌های موجود در نرم‌افزار اشاره می‌کند. گاهی اوقات بوی کد ناشی از طراحی و سبک برنامه‌نویسی ضعیف است. وجود بوی کد ممکن است باعث کندشدن پردازش، کاهش کیفیت کد، افزایش خطر خرابی برنامه و خطاهای نرم‌افزاری شود. تشخیص بوی کد یک چالش عمده در زمینه برنامه‌نویسی است و نشان می‌دهد که تشخیص و رفع آن یک مسئله پیچیده بوده و نیاز به روش‌ها و الگوریتم‌های مناسب و تطبیق‌پذیر دارد. تشخیص بوی کد بهبود کیفیت و قابلیت توسعه و اصلاح کد را فراهم می‌کند و از وقوع مشکلات و خطاها در آینده جلوگیری می‌نماید. برای شناسایی و برطرف‌کردن بوهای کد، محققان از الگوریتم‌های انتخاب ویژگی و یادگیری ماشین استفاده کرده‌اند. در این مقاله، برای انتخاب ویژگی‌های مفید و غیر افزونه، از میان ویژگی‌های متعدد نرم‌افزار، از الگوریتم فرااابتکاری گرگ خاکستری استفاده شده است. انتخاب ویژگی‌های مرتبط و مناسب، باعث سادگی مدل، بهبود دقت و کاهش زمان آموزش آن می‌شود. برای شناسایی و طبقه‌بندی بوهای کد، از ترکیب سه الگوریتم نزدیک‌ترین همسایه، درخت تصمیم و پشتیبان استفاده شده است. نتیجه نهایی طبقه‌بندی بر اساس سازوکار رأی‌گیری اکثریت تعیین می‌گردد. استفاده از ترکیب سه الگوریتم یادگیری ماشین و به‌کارگیری سازوکار رأی‌گیری اکثریت از نوآوری این مقاله محسوب می‌شود. جهت ارزیابی روش پیشنهادی، از مجموعه‌داده فونتانا که شامل بوهای کد متد طولانی، خصیصه حسادت، کلاس خدا و کلاس داده است، استفاده شده است. نتایج ارزیابی روش پیشنهادی بر اساس معیار دقت، برای شناسایی بوهای کد به ترتیب ۹۵، ۹۳،۵، ۹۳ و ۹۸،۵ درصد است که نسبت به مستقل پشتیبان با کرنل‌های مختلف، نزدیک‌ترین همسایه، درخت تصمیم و غیره عملکرد بهتری را نشان می‌دهد.

کلیدواژه‌ها: بوی رمز، انتخاب ویژگی، الگوریتم گرگ خاکستری، طبقه‌بندی، یادگیری ماشین، رأی‌گیری اکثریت.

۱- مقدمه

بوی کد[۱] به وضعیتی در کد منبع اشاره دارد که نشان‌دهنده وجود مشکلات و ضعف‌های احتمالی در طراحی و پیاده‌سازی آن است [۱]. این مشکلات می‌توانند توسط برنامه‌نویسان شناسایی شوند و با تحلیل و استفاده از فنون بهینه‌سازی کد، برطرف شوند. بوی کد می‌تواند به‌صورت متنوعی در کد منبع منعکس شود، مانند تکرار بیش از حد کد، پیچیدگی بالا، نام‌گذاری نامناسب، استفاده نادرست از الگوها و ساختارهای طراحی، عدم اصولی بودن متدها و کلاس‌ها و غیره [۲]. شناسایی و رفع بوی کد، قابلیت خوانایی، نگهداری و توسعه‌پذیری کد را بهبود می‌بخشد و درنهایت باعث افزایش کیفیت و عملکرد مناسب نرم‌افزار می‌شود [۳]. وجود بوی کد در توسعه نرم‌افزارهای بزرگ و پیچیده یک چالش واقعی است [۴]. آن‌ها پیامدهای اَعمال غیراستاندارد، محدودیت زمان پردازش و بی‌تجربگی طراحان نرم‌افزار هستند که مانع از ایجاد تغییرات بیشتر توسط توسعه‌دهندگان می‌شوند. نادیده‌گرفتن بوها منجر به خطاها و اشتباهات بزرگ‌تر در کد می‌شود [۵]. بوهای ناخوشایند می‌توانند شامل؛ رمزهای تکراری و مشابه، کلاس حجیم[۲]، کلاس تنبل[۳]، تعداد زیاد پارامترها[۱]،

Corresponding Author E-mail :mkarticles90@gmail.com

[1] Code Smell

[2] Large Classes

[3] Lazy Class

متدهای طولانی[۲] و غیره باشد [۶]. تشخیص بوی کد یک چالش در زمینه برنامه‌نویسی است که می‌تواند به‌صورت زیر مطرح شود:

۱- وجود بسیاری از زبان‌های برنامه‌نویسی و تفاوت‌های زبانی در ساختار و قواعد زبان می‌تواند تشخیص بوی کد را دشوار سازد. به‌عنوان‌مثال، بوی کد در یک زبان برنامه‌نویسی ممکن است با بوی کد در زبان دیگر متفاوت باشد.

۲- ساختار و قواعد زبان‌های برنامه‌نویسی ممکن است بسیار پیچیده باشد و تشخیص بوی کد را مشکل کند. برخی از زبان‌ها می‌توانند قواعد خاص و پیچیده‌ای برای ساختار کد داشته باشند که برای تشخیص بوی کد نیاز به تحلیل عمیق‌تری دارد.

۳- تنوع پروژه‌ها و سبک‌های برنامه‌نویسی می‌تواند تشخیص بوی کد را دشوار کند، زیرا بوی کد ممکن است بر اساس سبک شخصی برنامه‌نویس تغییر کند.

۴- تشخیص بوی کد ممکن است به درک مفاهیم و منطق کسب‌وکار پروژه نیاز داشته باشد. درک دقیق این مفاهیم و منطق می‌تواند برای تشخیص درست بوی کد بسیار مهم باشد.

تشخیص بوی کد اهمیت زیادی در توسعه نرم‌افزار دارد. ازجمله دلایل لزوم تشخیص بوی کد می‌توان به موارد زیر اشاره کرد. بهبود کیفیت کد، افزایش قابلیت خوانایی کد، کاهش پیچیدگی کد، افزایش قابلیت توسعه و اصلاح کد و کاهش احتمال وقوع خطاها. [۳].

هنگام مشاهده بوی کد، معمولاً اولین اقدام، رفع آن است. ابزارهای مختلف تشخیص بوی کد بر اساس تصویرسازی[۳] [۴]، رویکرد مبتنی بر ماشین[۴] [۵] و ارزیابی مبتنی بر معیار[۵] [۶] موجود هستند و ماهیت آن‌ها دستی، خودکار و نیمه‌خودکار است [۷]. مطالعات اخیر، محدودیت‌های مربوط به استفاده از این ابزارها را علی‌رغم عملکرد خوب آن‌ها، برجسته کرده‌اند [۸]. همچنین این آشکارسازها ازنظر نتایج و مقادیر آستانه با یکدیگر سازگار نیستند. مقادیر آستانه نقش بزرگی در شمارش بوها ایفا می‌کنند و موارد بودار را از موارد بدون بو متمایز می‌کنند [۹]. چالش دیگر تفسیر ذهنی از بوها توسط کارشناسان است [۸]. فنون یادگیری ماشین بر این محدودیت‌ها غلبه می‌کند، به این معنی که استفاده از قوانین بوی کد، تفسیر ذهنی و تنظیم مقادیر آستانه به‌عنوان الگوریتم‌های یادگیری ماشین آموزش می‌بینند و در شناسایی بوهای کد استفاده می‌شوند ازاین‌جهت رویکرد یادگیری ماشین باعث صرفه‌جویی در زمان و هزینه می‌شود [۱۰].

انتخاب ویژگی باعث می‌شود مدل ساده‌تر، تفسیر و اجرا شود. این کار باعث می‌شود دقت بهبود یابد و نیز موجب کاهش زمان آموزش می‌گردد. این امر خطر خطاهای داده را در حین اجرا نیز کاهش می‌دهد [۱۱].

در این تحقیق جهت بهبود دقت شناسایی بوهای کد، از الگوریتم تکاملی گرگ خاکستری به‌منظور انتخاب ویژگی‌های بهینه و مرتبط استفاده شده است. انتخاب ویژگی‌های مناسب و غیر افزونه، بر عملکرد مدل یادگیری و افزایش دقت آن تأثیر به سزایی دارد. همچنین در بخش طبقه‌بندی از ترکیب الگوریتم‌های طبقه‌بندی پایه نظیر درخت تصمیم، نزدیک‌ترین همسایه و پشتیبان استفاده شده است که نتیجه نهایی از ترکیب این سه طبقه‌بندی بر اساس رأی‌گیری اکثریت محاسبه می‌گردد. اجرای این فرایند جزء نوآوری مقاله محسوب می‌شود.

در ادامه و در بخش دوم، مروری بر ادبیات تحقیق ارائه خواهد شد، سپس در بخش سوم پیشینه تحقیق را مورد بررسی قرار می‌دهیم، در بخش چهارم نیز به ارائه روش پیشنهادی می‌پردازیم. در بخش پنجم ارزیابی روش پیشنهادی تشریح می‌شود و درنهایت، در بخش ششم نتیجه‌گیری و کارهای آینده مرتبط با این تحقیق بیان می‌گردد.

۲- مرور ادبیات تحقیق

در این قسمت لازم است با برخی از مفاهیم اولیه مرتبط با تحقیق جاری آشنا شویم. این مفاهیم شامل آشنایی با بوی کد و انواع آن، عملیات انتخاب ویژگی، عملیات طبقه‌بندی و طبقه‌بندهای مورداستفاده شامل: نزدیک‌ترین همسایه، پشتیبان، درخت تصمیم و همچنین الگوریتم تکاملی گرگ خاکستری (برای انتخاب ویژگی)، در ادامه موردبحث و بررسی قرار خواهند گرفت.

۲-۱- آشنایی با بوی کد

بوی کد نشان‌دهنده طراحی و پیاده‌سازی ضعیف برنامه است. بوهای کد معمولاً در سیستم‌های نرم‌افزاری مطرح می‌شوند. زیرا توسعه‌دهندگان، طراحی اجزای کد را به‌درستی درک نکرده یا به دلیل محدودیت‌های زمانی، به طراحی صحیح راه‌حل اهمیت نداده‌اند. بوی کد اگر شناسایی و برطرف نشود، ممکن است تأثیر منفی بر کیفیت نرم‌افزار، مانند قابلیت نگهداری، خوانایی کد و آزمون‌پذیری نرم‌افزار داشته باشد [۳، ۸، ۹].

۲-۱-۱- انواع بوی کد

برای درک چگونگی نقض اصول کدنویسی باکیفیت خوب، لازم است انواع بوی کد را بشناسیم. در این بخش به طور مختصر، بوهای پیشنهادشده توسط فاولر و بک [۲] شرح داده می‌شود.

[1] Long Parameter List
[2] Long Methods
[3] Illustration
[4] Machine Learning
[5] Metric

میکا مانتی لا [۱۲]، سپس بر اساس ویژگی‌های مشترک در بین بوهای کد، یک طبقه‌بندی خاصی را ارائه کرد تا برای مهندسان نرم‌افزار به‌وضوح قابل‌درک باشد. خلاصه‌ای از فهرست بوهای کد پیشنهادی توسط فاولر و بک در جدول (۱) نشان‌داده‌شده است. چهار نوع از مشهورترین و مهم‌ترین بوهای کد شامل Data Class، Long method، Feature Envy و God Class می‌باشد.

- Data Class: این بوی کد یک کلاس تماماً داده و بدون متد است. کلاسی است که به‌صورت غیرفعال و بدون عملکرد منطقی خاصی، تنها داده‌ها را ذخیره می‌کند.
- Long method: متدی است که دارای کد طولانی است و چندین وظیفه را به عهده دارد.
- Feature Envy: به متدهایی اشاره دارد که از داده‌ها و ویژگی‌های کلاس‌های دیگر، بیشتر از داده‌های کلاس خودشان استفاده می‌کنند.
- God Class: رده خدا به پیچیده بودن، داشتن رمز طولانی، استفاده از مقادیر زیادی از داده‌های کلاس‌های دیگر و اجرای چندین وظیفه مختلف تمایل دارد.

سپس میکا مانتیلا [۱۲]، بوهای کدی که توسط فاولر و بک شناسایی شده بودند را در هفت نوع مختلف دسته‌بندی نمود که این دسته‌بندی در جدول (۱) نشان‌داده‌شده است [۲].

جدول (۱): طبقه‌بندی بوهای کد [۲]

نوع بوی کد	طبقه‌بندی
Long Method Large Class	Bloaters
Switch Statements Temporary Fields	Oriented Object Abusers
Divergent Change Shotgun Surgery	Change Preventers
Lazy Class Data Class	Dispensables
Message Chain Middle Man	Encapsulators
Feature Envy Inappropriate Intimacy	Couplers
Incomplete Library Class Comments	Others

برای بهبود کیفیت نرم‌افزار در درازمدت، لازم است بوهای کد به‌دقت بررسی و رفع شوند. رفع‌شدن آنها ریسک انباشته‌شدن بوی کد را در پروژه کم می‌کند. یکی از فواید جلوگیری از انباشته‌شدن چنین الگوهای بد در پروژه‌های نرم‌افزاری، بهبود فرایند نگهداشت آن است که موضوعی بسیار مهم برای چابکی گروه مجهز نگهداری نرم‌افزار محسوب می‌شود.

۲-۲- انتخاب ویژگی

معمولاً برای ساخت مدل‌های شناسایی بوی کد از فنون مختلف یادگیری ماشین استفاده می‌شود که طبق تحقیقات انجام‌شده، عملکرد این فنون بسیار وابسته به مجموعه‌داده یادگیری و آموزشی است. مجموعه‌داده آموزشی معمولاً دارای ویژگی‌های نرم‌افزاری زیادی است که برخی از آن‌ها نامرتبط و یا افزونه است که باعث کاهش دقت مدل طبقه‌بند می‌شوند. حذف این ویژگی‌های نامرتبط با استفاده از انتخاب ویژگی انجام می‌گیرد. در واقع انتخاب ویژگی، به‌منظور حذف ویژگی‌های غیرضروری، کاهش ابعاد داده‌ها، کاهش هزینه‌های محاسباتی و افزایش دقت مدل یادگیری ماشین مورداستفاده قرار می‌گیرد. به‌عبارت‌دیگر، انتخاب ویژگی برای پاک‌سازی داده‌های نویزی، اضافی و غیرمرتبط استفاده می‌شود. درنتیجه، سرعت یادگیری و عملکرد مدل طبقه‌بندی افزایش می‌یابد [۱۳].

۲-۲-۱- انواع انتخاب ویژگی

- **روش فیلتر:** در این روش، ویژگی‌ها بر اساس اندازه‌گیری‌های آماری انتخاب می‌شوند. این روش مستقل از الگوریتم یادگیری است و به زمان محاسباتی کمتری نیاز دارد.
- **روش پوشش:** عملکرد روش پوشش بستگی به نوع طبقه‌بندی‌کننده دارد. بهترین زیرمجموعه ویژگی‌ها بر اساس نتایج طبقه‌بندی‌کننده انتخاب می‌شود. پوشش، به دلیل گام‌های یادگیری تکراری و اعتبارسنجی متقابل، ازنظر محاسباتی بسیار گران‌تر از فیلتر هستند. بااین‌حال، این روش‌ها دقیق‌تر از روش فیلتر هستند.

روش ترکیبی:

رویکرد سوم روش ترکیبی است که از یادگیری جمعی و یادگیری ترکیبی برای انتخاب ویژگی استفاده می‌کند. ازآنجاکه این روش یک تصمیم جمعی است، عملکرد آن نسبت به دو مدل دیگر بهتر است. جنگل تصادفی یکی از این نمونه‌ها است. ازنظر محاسباتی فشرده‌سازی کمتری نسبت به پوشش دارد. بااین‌حال، این روش نیز دارای اشکال خاصی برای مدل یادگیری است [۱۳].

۲-۳- طبقه‌بندی

فنون داده‌کاوی به طور گسترده به دودسته: پیش‌بینی‌کننده و توصیفی دسته‌بندی می‌شوند. هر دوی این روش‌ها برای استخراج الگوهای پنهان از حجم عظیمی از داده‌ها استفاده می‌شود. طبقه‌بندی در واقع، فرایند تبدیل سوابق داده به مجموعه‌ای از کلاس‌ها است.

طبقه‌بندی به دو قسمت زیر تقسیم می‌شوند:

- طبقه‌بندی تحت نظارت
- طبقه‌بندی بدون نظارت

در طبقه‌بندی تحت نظارت، داده‌هایی که باید طبقه‌بندی شوند قبلاً بر اساس چند فرضیه شناخته‌شده‌اند. در طبقه‌بندی بدون نظارت، مجموعه موارد توسط کاربران قابل پیش پیش‌بینی‌نشده است. با این فرض که این وظیفه کاربر است که داده‌های ارائه‌شده را طبقه‌بندی کرده و برای تعیین نام آن کلاس‌ها تلاش کند. این نوع طبقه‌بندی به‌عنوان خوشه‌بندی شناخته می‌شود. طبقه‌بندی شامل پیش‌بینی یک نتیجه خاص بر اساس یک ورودی خاص است. به‌منظور پیش‌بینی نتایج، باید داده‌های موجود را از قبل تهیه کرد. بر اساس این داده‌ها، کلاس‌ها طبقه‌بندی می‌شوند.

شکل (۱): یک طبقه‌بندی از **x** به**y** [۱۴]

شکل (۱) سازوکار عملیات طبقه‌بندی را نشان می‌دهد. الگوریتم‌های طبقه‌بندی استفاده‌شده در این تحقیق شامل موارد زیر است:

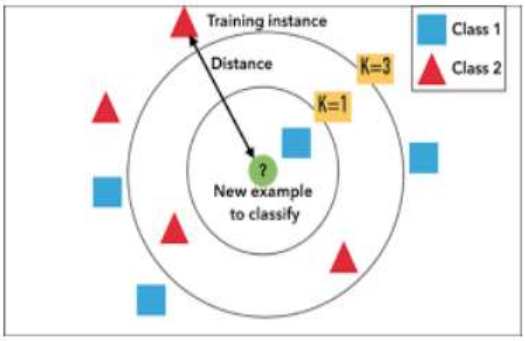
- نزدیک‌ترین همسایه
- درخت تصمیم
- پشتیبان

۲-۳-۱- الگوریتم نزدیک‌ترین همسایه

الگوریتم K-نزدیک‌ترین همسایه[1] یک روش برای طبقه‌بندی مبتنی بر نزدیک‌ترین اشیا آموزشی در فضای ویژگی است. قانون نزدیک‌ترین همسایه (NN) یک روش تصمیم‌گیری شناخته‌شده و غیرپارامتری برای یادگیری ماشین و داده‌کاوی است. قواعد رأی‌گیری k-نزدیک‌ترین همسایه بر اساس مقدار K (k> 1) محاسبه می‌شود. مطابق شکل (۲) کلاس پیش‌بینی‌شده x مجموعه‌ای است برابر باکلاسی که اکثریت نزدیک‌ترین همسایگان را در مجموعه آموزشی نشان می‌دهد.

جستجوی K نزدیک‌ترین همسایه، K همسایه نزدیک‌تر به نقطه پرس‌وجو را برمی‌گرداند. این روش معمولاً در تجزیه‌وتحلیل پیش‌بینی، به‌منظور تخمین و یا دسته‌بندی یک نقطه بر اساس اجماع همسایگان آن استفاده می‌شود درواقع ایده اصلی این است که یک ورودی با توجه به K نزدیک‌ترین ورودی‌های دیگر طبقه‌بندی می‌شود. فرض کنید n داده آموزشی با دانستن کلاس هرکدام در اختیارداریم. می‌خواهیم یک بردار ویژگی نامعلوم x را دریکی از c کلاس فوق دسته‌بندی کنیم. ابتدا معیاری را برای اندازه‌گیری فاصله در نظر می‌گیریم (فاصله اقلیدسی) سپس به روش زیر عمل می‌کنیم [۱۵]:

- فاصله x را با هر بردار از داده‌های آموزشی حساب می‌کنیم؛ بدون درنظرگرفتن کلاس آن k داده که نزدیک‌ترین فاصله را تا x دارند انتخاب می‌کنیم. مقدار k فرد انتخاب می‌گردد.
- از این k نمونه، تعداد اعضای هر کلاس را مشخص می‌کنیم و تعداد اعضای ω_i از این k نمونه را k_i می‌نامیم. (i=1,…,c)
- X را در کلاسی دسته‌بندی می‌کنیم که بزرگ‌ترین k_i را داشته باشد.



شکل (۲): طبقه‌بندی بر اساس k- نزدیک‌ترین همسایگان [۱۶]

شکل (۲) سازوکار عملیات طبقه‌بندی **k**- نزدیک‌ترین همسایگان را نشان می‌دهد.

۲-۳-۲- درخت تصمیم‌گیری

درخت تصمیم[2] یک ابزار برای پشتیبانی از تصمیم است که از درخت‌ها برای مدل کردن استفاده می‌کند. درخت تصمیم‌گیری، یک نمودار ترسیمی است که روند تصمیم‌گیری یا سلسله‌ای از تصمیمات را نمایش می‌دهد. یک ابزار کمکی برای تصمیم‌گیری است و از یک گراف یا مدل درختی شکل که تصمیمات و عواقب محتمل آن‌ها را نمایش می‌دهد، تشکیل‌شده است. درخت تصمیم‌گیری، تصمیمات دشوار را به تصمیمات کوچک‌تر، ساده‌تر و منطقی‌تر تقسیم می‌کند. به این صورت می‌تواند همه‌ی افراد را در تصمیمات سخت یاری دهد. درخت تصمیم ساختاری است که

[1] K-Nearest Neighbors Algorithm (knn)

[2] Decision Tree

data pointها را انشعاب‌بندی می‌کند یا آن‌ها را طبقه‌بندی می‌نماید. حسن این درخت‌ها این است که تفسیر آن‌ها راحت است و به‌راحتی می‌توان فهمید که چه سوالی پرسیده شده و داده بر چه اساسی تقسیم‌بندی شده است [۱۷]. مفاهیم اصلی در درخت‌های تصمیم شامل موارد زیر است:

- گره: به‌منظور مستقل که آزمون روی آن انجام می‌شود.
- گره ریشه: گره‌ای که در بالاترین نقطه درخت وجود دارد.
- برگ: به متغیر وابسته یا برچسب دسته، برگ می‌گویند.
- شاخه: به مقیاسی که خروجی از آن تعیین می‌شود می‌گویند.

۳-۳-۲- پشتیبان

پشتیبان یک روش قدرتمند برای ایجاد طبقه‌بندی است. هدف آن ایجاد یک مرز تصمیم‌گیری بین دو کلاس است که امکان پیش‌بینی برچسب‌ها را از یک یا چند بردار ویژگی فراهم می‌کند. این مرز تصمیم‌گیری که به‌عنوان ابر صفحه شناخته می‌شود، به‌گونه‌ای طراحی‌شده است که تا جایی که ممکن است از نزدیک‌ترین نقطه داده‌ها از هر یک از کلاس‌ها استفاده شود. این نقاط نزدیک، بردارهای حمایتی نامیده می‌شوند [۱۸].

۳- پیشینه تحقیق

در این بخش، به تشریح تحقیقات انجام‌گرفته توسط سایر پژوهشگران در زمینه تشخیص بوی کد می‌پردازیم. هر یک از این کارها در بخش‌های جداگانه به‌صورت جامع شرح داده شده است.

کیاک و همکاران [۱۹]از فنون یادگیری ماشین، به‌ویژه طبقه‌بندی چند برچسبی[1]، برای طبقه‌بندی اینکه آیا کد منبع داده شده با بیش از یک کد، دارای بوی کد است یا خیر، استفاده کرده‌اند. دو روش طبقه‌بندی چند برچسبی (زنجیره‌های طبقه‌بندی‌کننده و ترکیب برچسب‌ها) و مدل‌های مجموعه آن‌ها بر روی مجموعه‌داده‌های تبدیل‌شده با استفاده از پنج طبقه‌بندی‌کننده پایه مختلف انجام شد. نتایج نشان داد که به‌عنوان طبقه‌بندی‌کننده پایه، الگوریتم جنگل تصادفی عملکرد بهتری نسبت به الگوریتم درخت تصمیم و نایوبیز دارد.

پکورلی و همکاران [۲۰]یک مقایسه تجربی در مقیاس بزرگ بین پنج روش مختلف متعادل‌سازی داده برای تشخیص بوی کد مبتنی بر یادگیری ماشین ارائه کردند. هدف از این مطالعه درک تأثیر فنون متعادل‌سازی داده‌ها بر عملکرد الگوریتم‌های یادگیری ماشین در تشخیص بوی کد است. نتایج نشان داد که مدل‌های یادگیری ماشین باتکیه‌بر روش متعادل‌سازی SMOTE[2] بهترین عملکرد را دارند.

پکورلی و همکاران [۲۱]یک مطالعه در مقیاس بزرگ را برای مقایسه تجربی عملکرد فنون مبتنی بر الگوریتم‌های فرااکتشافی و یادگیری ماشین برای تشخیص بوی کد پیشنهاد کردند. آن‌ها پنج نوع بوی کد را در نظر گرفتند و مدل‌های یادگیری ماشین را با DECOR[3] مقایسه کردند. یافته‌های کلیدی بر نیاز به تحقیقات بیشتر باهدف بهبود اثربخشی هر دو روش یادگیری ماشین و فرااکتشافی برای تشخیص بوی کد تأکید دارند، درحالی‌که DECOR به‌طورکلی عملکرد بهتری نسبت به پایه یادگیری ماشین دارد. همچنین دقت آن بسیار پایین است تا آن را در عمل قابل‌استفاده کند.

ابراهیم و همکاران [۲۲]روشی را در مورد چگونگی تشخیص و بازآرایی[4] بوی کد از رمزهای منبع یک برنامه اندرویدی ارائه دادند تا از افزونگی در تولید داده‌های آزمون کاسته شود. نتایج نشان داد که DART[5] قادر است افزونگی را در داده‌های آزمون کاهش دهد، درحالی‌که تمایل به بهبود کارایی داده‌های آزمون ایجادشده وجود دارد.

گوگول وتو و همکاران [۲۳]، از طبقه‌بندی چند برچسبی برای تشخیص اینکه آیا کد منبع داده شده دارای بوهای کد متعددی است یا خیر، استفاده کرده‌اند. آن‌ها دو مجموعه‌داده بوی کد برای این کار در نظر گرفته‌اند و آن‌ها را به مجموعه‌داده‌های چند برچسبی تبدیل کرده‌اند. در آزمون آن‌ها، دو روش چند برچسبی روی مجموعه‌داده تبدیل‌شده، عملکردهای خوبی در اعتبارسنجی متقابل ۱۰ برابر، با استفاده از ده تکرار، نشان می‌دهد.

جین و همکاران [۲۴]، از فن انتخاب ویژگی ترکیبی با الگوریتم‌های یادگیری ماشین برای بهبود عملکرد در تشخیص بوهای کد استفاده کردند. نتایج نشان داد که جنگل تصادفی و رگرسیون لجستیک بهترین طبقه‌بندی‌کننده‌های یادگیری ماشین هستند. بوی کد «Data Class» قابل‌تشخیص‌ترین بوی کد بود. روش پشته‌سازی[6]، همیشه در مقایسه با طبقه‌بندهای جداگانه، نتایج بهتری می‌دهد.

در جمع‌بندی این بخش، می‌توان گفت که کارهای تحقیقاتی که اخیراً در حوزه شناسایی بوهای کد انجام شده است موردمطالعه و بررسی قرار گرفت. هدف از این مطالعه، بیان اهداف، رویکردها، مزایا و نقاط ضعف هر یک از این کارها می‌باشد. از مطالعات

[1] Multi-Labeling Classification

[2] روشی برای ساخت طبقه بندها از مجموعه داده‌های نامتعادل

[3] روشی برای ارزیابی الگوریتم‌های یادگیری ماشین

[4] Refactoring

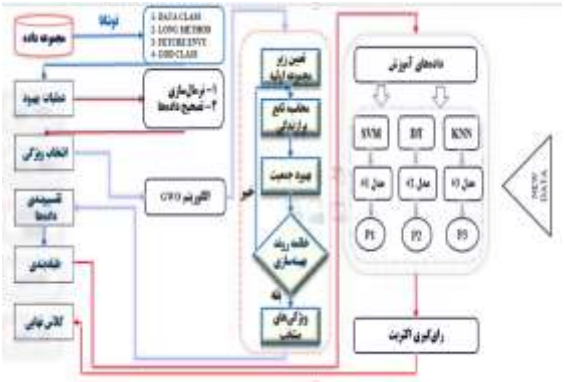
[5] Detection and Refactoring Tool

[6] Stacking

انجام‌شده، دریافتیم که اولاً حوزه تشخیص بوهای کد در سامانه‌های نرم‌افزاری، یک‌روند تحقیقاتی مهم است و ثانیاً با متعدد محققین مختلف برای افزایش دقت شناسایی بوی کد آشنا شدیم. همچنین، در این بخش جهت افزایش دقت شناسایی بوی کد، به سمت انتخاب ویژگی سوق داده شدیم. زیرا با بررسی تحقیقات مرتبط، متوجه شدیم که تعدد ویژگی‌ها تأثیر مستقیم و منفی بر عملکرد الگوریتم‌های شناسایی بوی کد دارند. در ادامه، با مطالعه و بررسی پیشینه تحقیق، به رهیافتی مبنی بر به‌کارگیری الگوریتم‌های فراابتکاری/تکاملی برای انتخاب ویژگی و استفاده از الگوریتم‌های یادگیری ماشین جهت ساخت مدل طبقه‌بندی (تشخیص‌دهنده بوی کد) دست یافتیم. در واقع، یاد گرفتیم که برای ارائه روش پیشنهادی خود در این مقاله به‌منظور افزایش دقت شناسایی بوهای کد، به دو مسئله انتخاب ویژگی و الگوریتم‌های یادگیری ماشین توجه ویژه‌ای معطوف کنیم. البته مقاله ما در روش انتخاب ویژگی و ساخت مدل تشخیص‌دهنده بوی کد، دارای نوآوری خاص خود است و ازاین‌جهت با مقالات دیگر کاملاً متمایز می‌باشد. گفتنی است، مقالاتی که موردمطالعه و بررسی قرار گرفته‌اند، بسیار بیش از آن چیزی است که به طور مختصر در این بخش آورده شده است.

۴- روش پیشنهادی

در این مقاله، جهت تشخیص بوی کد از یک سازوکار چندمرحله‌ای استفاده شده است که مراحل آن در ادامه شرح شده است.



شکل (۳): سازوکار روش پیشنهادی جهت شناسایی بوی کد

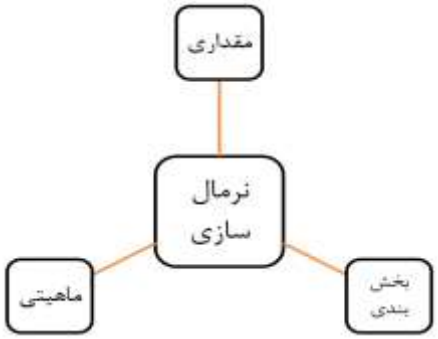
همان‌طور که در شکل (۳) نشان داده شده است، جهت تشخیص بوی کد از یک سازوکار چندمرحله‌ای بهره گرفته شده است. در مرحله نخست پس از انتخاب دیتاست مناسب جهت تشخیص بوی کد، عملیات بهبود انجام می‌گیرد. عملیات بهبود به مراحل نرمال‌سازی و تصحیح داده‌ها وابسته است که هر یک از این مراحل باعث تغییر داده‌ها می‌گردد. در گام دوم، عملیات انتخاب ویژگی انجام می‌گیرد که این عملیات توسط الگوریتم تکاملی گرگ خاکستری انجام می‌شود و بعدازآن، مرحله ساخت مدل آموزشی انجام می‌گیرد. در این بخش، ویژگی‌های انتخابی به الگوریتم‌های طبقه‌بندی داده می‌شود تا عملیات ساخت مدل انجام گیرد. بعد از ایجاد مدل، از داده‌های آزمون جهت ارزیابی مدل استفاده می‌گردد که نتایج آن توسط سازوکار رأی‌گیری اکثریت به دست می‌آید. در ادامه، هر یک از بخش‌های موردنظر شرح داده خواهد شد.

۴-۱- عملیات بهبود

همان‌طور که در بخش پیشین نیز اشاره گردید، وظیفهٔ این مرحله تغییر و متعادل‌سازی داده‌ها است. جهت تغییر داده‌ها از یک سازوکار سه‌مرحله‌ای بهره گرفته‌شده است که در ادامه، شرح داده خواهد شد.

- **نرمال‌سازی داده‌ها**

یکی از بخش‌های مهم روش بهبود، مربوط به بخش نرمال‌سازی داده‌ها است. نرمال‌سازی داده‌ها، به‌منظور تغییر ماهیت داده‌ها مورداستفاده قرار می‌گیرد. نرمال‌سازی داده‌ها می‌تواند توسط سه سازوکار نشان‌داده‌شده در شکل (۴) مورداستفاده قرار گیرد.



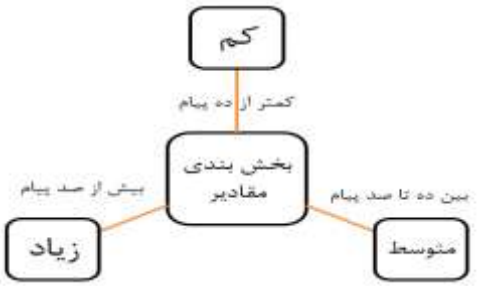
شکل (۴): سازوکار عملیات نرمال‌سازی

همان‌طور که در شکل (۴) مشاهده می‌شود، عملیات نرمال‌سازی در سه گروه مقداری، ماهیتی و بخش‌بندی انجام می‌گیرد. در بخش مقداری، مقادیر ویژگی‌ها از یک بازه‌ی [a,b] به یک بازه‌ی [c,d] تغییر می‌کند که جهت عملیات تغییر از رابطه‌ی (۱) بهره گرفته می‌شود.

$$N = \frac{F-a}{b-a} * (d-c) + c \qquad \text{(رابطه ۱)}$$

در رابطهٔ (۱)، N نشان‌دهنده مقدار ویژگی بعد از انجام عملیات نرمال‌سازی، F نشان‌دهنده مقدار ویژگی قبل از انجام عملیات نرمال‌سازی، a نشان‌دهنده حداکثر مقدار ویژگی قبل از انجام عملیات نرمال‌سازی، b نشان‌دهنده حداقل مقدار ویژگی قبل از انجام عملیات نرمال‌سازی، c نشان‌دهنده حداقل مقدار ویژگی بعد از انجام عملیات نرمال‌سازی و d نشان‌دهنده حداکثر مقدار ویژگی بعد از انجام عملیات نرمال‌سازی است. در عملیات بخش‌بندی مقادیر

ویژگی‌ها به بخش‌های مختلف تقسیم‌بندی می‌گردد. به عبارتی در این حالت بر روی مقادیر ویژگی‌ها عملیات گسسته سازی صورت می‌گیرد. عملیات گسسته سازی در سه بخش مقدار کم، متوسط و زیاد صورت می‌گیرد. به‌عنوان‌مثال، برای ویژگی ارسال بسته‌ها، می‌توان مقادیر ویژگی‌ها را به سه بازه‌ی نشان داده‌شده در شکل (۵) دسته‌بندی نمود.

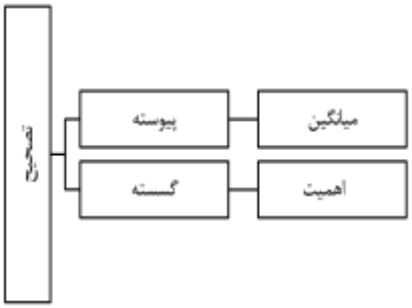


شکل (۵): سازوکار بخش‌بندی مقادیر ویژگی‌ها

در بخش ویژگی‌های ماهیتی، مقادیر ویژگی‌ها از ماهیت اسمی به ماهیت عددی تغییر می‌کنند. جهت تغییر ماهیت ویژگی‌ها، به هر ویژگی اسمی، یک مقدار واحد اختصاص داده می‌شود که اعداد موردنظر نشان‌دهنده مقادیر ویژگی است. به‌عنوان‌مثال، برای ویژگی پروتکل انتقال که شامل مقادیر UDP، FTP و TCP است به ترتیب رمزهای ۱، ۲ و ۳ در نظر گرفته می‌شود.

- **تصحیح داده**

هدف از عملیات تصحیح داده، تغییر مقادیر داده است. در این مقاله، عملیات تصحیح داده به داده‌های مفقودی مربوط می‌گردد. داده‌های مفقودی به داده‌هایی گفته می‌شود که مقادیر آن‌ها در مجموعه‌داده به‌عنوان دادهٔ Null شناخته می‌شود. جهت تصحیح این نوع مقادیر از روش نشان‌داده‌شده در شکل (۶) استفاده می‌شود.



شکل (۶): سازوکار تصحیح داده

همان‌طور که در شکل (۶) مشاهده می‌شود، جهت تصحیح داده‌ها از دو سازوکار میانگین و مقدار پراهمیت استفاده‌شده است. وقتی از سازوکار میانگین استفاده می‌شود که نوع ویژگی نشان‌دهنده پیوستگی باشد. در این حالت از رابطه‌ی (۲) جهت عملیات تصحیح داده استفاده می‌گردد.

$$X_{Miss} = \sum_{i \epsilon idx_{nonmiss}} X_F \qquad \text{(رابطه۲)}$$

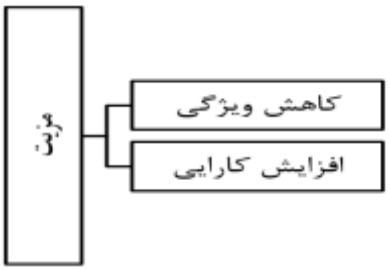
در رابطهٔ (۲)، X_{Miss} نشان‌دهنده مقادیری از نمونه‌ها است که دارای مقدار Null است. X_F نشان‌دهنده مقادیری از نمونه‌ها است که مقادیر آن‌ها عدد صحیح است و $idx_{nonmiss}$ نشان‌دهنده اندیس نمونه‌هایی است که دارای مقادیر غیر Null است. هنگامی‌که نوع ویژگی نشان‌دهنده گسستگی ویژگی باشد، بجای عملیات میانگین از فن پراهمیت‌ترین مقدار استفاده می‌گردد. در این حالت بجای مقادیر Null از مقداری استفاده می‌شود که بیشترین تکرار را داشته باشد. در رابطه‌ی (۳) این عملیات نشان داده‌شده است.

$$X_{Miss} = Arg \max_{j \in D} \sum X_j \qquad \text{(رابطه۳)}$$

در رابطهٔ (۳)، X_j نشان‌دهنده نمونه‌هایی است که دارای مقدار D است. D نشان‌دهنده دسته‌های مجزا در ویژگی iام است.

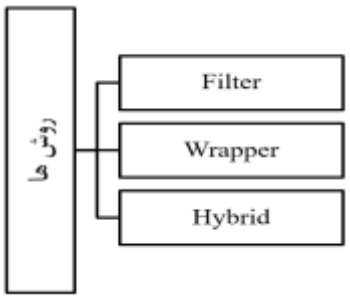
۴-۲- انتخاب زیرمجموعه بهینه

هدف از این بخش، انتخاب زیرمجموعه مناسب از ویژگی‌ها است. برای مثال، فرض کنید یک مجموعه‌داده به تعداد d ویژگی در دسترس باشد. عملیات انتخاب ویژگی جهت انتخاب S ویژگی از میان این d ویژگی است، به‌طوری‌که ویژگی‌های انتخابی خواص نشان‌داده‌شده در شکل (۸) را داشته باشد.



شکل (۸): مزایای زیرمجموعه انتخابی

همان‌طور که در شکل (۸) نشان داده‌شده است، زیرمجموعه انتخابی باید بتواند علاوه بر کاهش ویژگی‌ها، باعث افزایش دقت مدل تشخیص بوی کد نیز بشود. جهت انتخاب یک زیرمجموعه S-عضوی از میان یک مجموعه d-عضوی، از نشان داده‌شده در شکل (۹) بهره گرفته می‌شود.

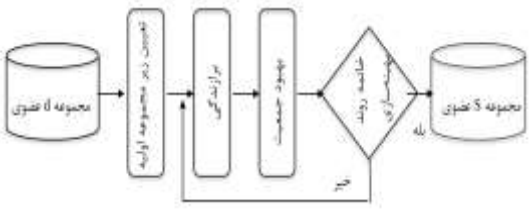


شکل (۹): سازوکار انتخاب زیرمجموعه S-عضوی

همان‌طور که در شکل (۹) نشان داده‌شده است، جهت انتخاب یک مجموعه S-عضوی از میان یک مجموعه d-عضوی از سه روش با ناظر، بدون ناظر و ترکیبی استفاده می‌شود که روش با ناظر به‌عنوان پوشش شناخته می‌شود و بدون ناظر به‌عنوان فیلتر شناخته می‌شود. در این مقاله، از پوشش جهت انتخاب زیرمجموعه S-عضوی استفاده می‌گردد. مهم‌ترین فن مورداستفاده در پوشش، بهره‌گیری از الگوریتم‌های تکاملی است. ازاین‌رو، در این مقاله از الگوریتم تکاملی گرگ خاکستری جهت انتخاب زیرمجموعه S-عضوی بهره گرفته‌شده است. در ادامه توضیحات مربوط به این الگوریتم آورده شده است.

۴-۳- انتخاب زیرمجموعه بهینه با استفاده از الگوریتم گرگ خاکستری

الگوریتم گرگ خاکستری، جهت حل مسئلهٔ انتخاب زیرمجموعه بهینه از فرایندهای نشان‌داده‌شده در شکل (۱۰) بهره می‌گیرد که این مراحل به شرح زیر آورده شده است.



شکل (۱۰): سازوکار انتخاب زیرمجموعه S-عضوی توسط الگوریتم گرگ خاکستری

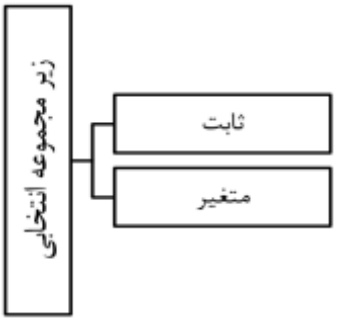
همان‌طور که در شکل (۱۰) مشاهده می‌شود، فرایند انتخاب زیرمجموعه S-عضوی توسط الگوریتم گرگ خاکستری توسط یک مرحله تولید و دو مرحله تکرارشونده تشکیل‌شده است. در مرحله نخست، زیرمجموعه اولیه تولید می‌شوند که این مرحله یک‌بار در ابتدای الگوریتم صورت می‌گیرد. بعد از تولید زیرمجموعه اولیه، دو فرایند تکراری که شامل محاسبه‌ی برتری زیرمجموعه (محاسبه تابع برازندگی) و بهبود زیرمجموعه است، انجام می‌گیرد. این عملیات تا زمانی ادامه پیدا می‌کند که شرط خاتمه‌ی الگوریتم نقض نگردد. قبل از بیان بخش‌های مختلف الگوریتم گرگ خاکستری، در ابتدا لازم است درباره‌ی اصطلاح گرگ و طعمه و نحوه‌ی نگاشت این دو پارامتر به مسئله‌ی انتخاب زیرمجموعه توضیح داده شود.

جمعیت اولیه: در این مقاله، جمعیت اولیه توسط تعداد گرگ‌ها نشان داده می‌شود. بر اساس مطالب قبل، هر جمعیت اولیه نشان‌دهنده یک زیرمجموعه S-عضوی است؛ بنابراین، هر گرگ نشان‌دهنده یک عضو از زیرمجموعه انتخابی است.

طعمه: در الگوریتم گرگ خاکستری، برای نزدیکی گرگ‌ها به طعمه از معیار برازندگی استفاده می‌شود. به عبارتی، هر گرگی که مقدار برازندگی آن بهتر باشد، به طعمه نزدیک‌تر است؛ بنابراین، طعمه در الگوریتم گرگ خاکستری، نشان‌دهنده برتری یک گرگ نسبت به گرگ دیگر است.

- **تعیین زیرمجموعه اولیه**

هدف از این بخش، تعیین زیرمجموعه‌های تصادفی جهت شروع الگوریتم گرگ خاکستری است. زیرمجموعه‌های انتخابی به دو صورت می‌توانند تولید شوند که در شکل (۱۱) نشان داده‌شده است.



شکل (۱۱): تعیین زیرمجموعه انتخابی

همان‌طور که در شکل (۱۱) نشان داده‌شده است، زیرمجموعه انتخابی می‌تواند در دو حالت ثابت و متغیر مورداستفاده قرار گیرد. در حالت ثابت، طول همه‌ی جمعیت‌ها ثابت است. در این حالت، برای انتخاب یک مجموعه از مجموعه دیگر، از باینری‌سازی استفاده می‌گردد. در حالت متغیر، طول هر جمعیت نسبت به جمعیت دیگر متفاوت است. در حالت متغیر، هر جمعیت نشان‌دهنده اندیس عضوهای انتخابی است. به‌عنوان‌مثال، برای یک مجموعه ویژگی ۵ عضوی، جمعیت ثابت و متغیر می‌تواند به‌صورت شکل (۱۲) باشد.

شکل (۱۲): سازوکار تعیین زیرمجموعه انتخابی ثابت و متغیر

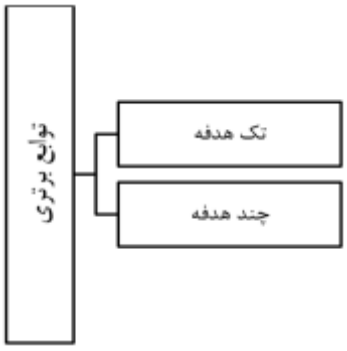
همان‌طور که در شکل (۱۲) نشان داده‌شده است، هر دو زیرمجموعه انتخابی، عضوهای دوم، چهارم و پنج را انتخاب کرده‌اند. در حالت ثابت، مصرف حافظه زیاد ولی پیچیدگی مسئله کم است. در حالت متغیر، مصرف حافظه کاهش پیدا می‌کند ولی پیچیدگی مسئله افزایش پیدا می‌کند. ازاین‌رو، در این مقاله از روش ثابت استفاده‌شده است. همان‌طور که در بخش اولیه‌ی این مبحث بیان گردید، برای بهره‌گیری از زیرمجموعه انتخابی در حالت ثابت، باید جمعیت موردنظر باینری گردد. جهت باینری کردن جمعیت از رابطه‌ی (۴) استفاده می‌شود.

$$binary\ Select = \begin{cases} Feature\ Select & if\ p_{ij} \geq 0.5 \\ Non\ Feature\ Select & if\ p_{ij} < 0.5 \end{cases} \quad \text{(رابطه۴)}$$

در رابطهٔ بالا، $binary\ Select$ نشان‌دهنده عملیات انتخاب عضو و p_{ij} نشان‌دهنده مقدار عضو jام برای جمعیت iام است. اگر مقدار این عضو بیشتر از ۰٫۵ در نظر گرفته شود، عضو موردنظر به‌عنوان عضو انتخابی شناخته می‌شود ولی اگر مقدار عضو کمتر از ۰٫۵ در نظر گرفته شود، این عضو به‌عنوان عدم انتخاب در نظر گرفته می‌شود.

- **تعیین برتری مجموعه اولیه**

برای نشان‌دادن برتری یک زیرمجموعه انتخابی نسبت به یک زیرمجموعه انتخابی دیگر، از تابع برازندگی استفاده می‌شود. تابع برازندگی نشان‌دهنده قدرت عضوهای انتخابی است. تابع برازندگی می‌تواند در دو حالت نشان‌داده‌شده در شکل (۱۳) مورداستفاده قرار گیرد.



شکل (۱۳): تابع برازندگی از نظر کاربرد

همان‌طور که در شکل (۱۳) نشان داده‌شده است، برتری یک زیرمجموعه نسبت به زیرمجموعه دیگر می‌تواند در دو حالت تک هدفِ و چند هدفِ مورد مقایسه قرار گیرد. در حالت تک هدفِ، فقط یک هدف برای بهینه‌سازی وجود دارد، درحالی‌که در حالت چند هدفِ، چندین هدف برای بهینه‌سازی وجود دارد که این اهداف می‌تواند در تضاد با یکدیگر باشند. در این مقاله، از روش تک هدفِ استفاده‌شده است. تابع برازندگی روش پیشنهادی در ادامه آورده شده است.

$$Fitness_i = \frac{Mistake}{Total} * 100 \quad \text{(رابطه ۵)}$$

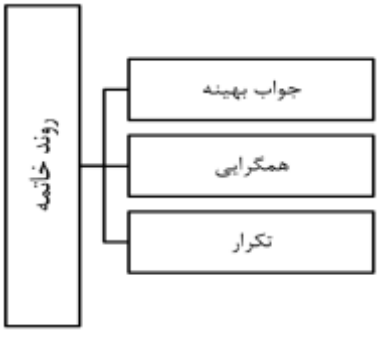
در رابطهٔ (۵) $Fitness_i$ نشان‌دهنده برازندگی زیرمجموعه انتخابی iام، $Mistake$ بیانگر تعداد نمونه‌هایی است که به‌اشتباه تشخیص داده‌شده است و $Total$ بیانگر کل نمونه‌ها می‌باشد.

- **بهبود زیرمجموعه اولیه**

الگوریتم گرگ خاکستری جهت بهبود زیرمجموعه اولیه از موقعیت گرگ‌های برتر استفاده می‌کند. به عبارتی، در این حالت گرگ‌ها به سمت گرگ‌های برتر حرکت می‌کنند. حرکت گرگ‌ها به سمت گرگ‌های برتر، نشان‌دهنده هماهنگ‌شدن عضوهای انتخابی با عضوهای رهبر است. در الگوریتم گرگ خاکستری، از موقعیت سه زیرمجموعه انتخابی برتر که شامل گرگ‌های آلفا، بتا و دلتا است، جهت بهبود زیرمجموعه انتخابی استفاده می‌شود. در این حالت، برآیندی از زیرمجموعه جاری با زیرمجموعه‌های انتخابی به دست می‌آید که از این برآیند برای بهبود جمعیت استفاده می‌گردد.

- **شرط توقف**

به‌منظور خاتمهٔ روند آموزش جهت یافتن زیرمجموعه بهینه، از نشان‌داده‌شده در شکل (۱۴) بهره گرفته می‌شود که در ادامه آورده شده است.



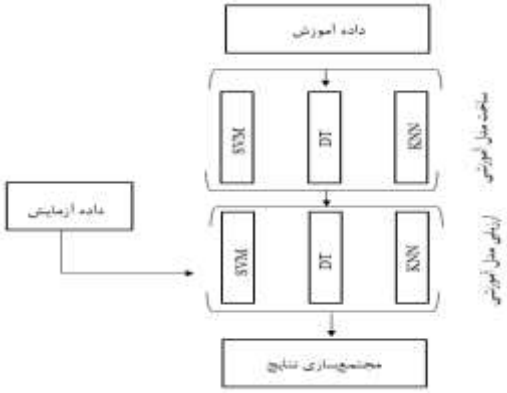
شکل (۱۴): سازوکار خاتمه الگوریتم گرگ خاکستری

همان‌طور که در شکل (۱۴) نشان داده‌شده است، جهت خاتمه روند آموزشی از سه روش جواب بهینه، عدم بهبود و تکرار استفاده می‌گردد. در حالت جواب بهینه، الگوریتم زمانی خاتمه می‌یابد که به جواب بهینه رسیده باشد. در حالت همگرایی، الگوریتم زمانی خاتمه می‌یابد که نتواند برای h دور خود را بهینه نماید و درنهایت در حالت تکرار، الگوریتم زمانی خاتمه می‌یابد

که به تعداد تکرار مشخص رسیده باشد.

۴-۴- ساخت مدل آموزشی

بعد از عملیات استخراج ویژگی و تعیین ویژگی‌های انتخابی، از الگوریتم‌های طبقه‌بندی جهت شناسایی بوی کد استفاده می‌شود. در این مقاله، از الگوریتم‌های طبقه‌بندی نزدیک‌ترین همسایه، درخت تصمیم و پشتیبان جهت ساخت مدل آموزشی استفاده‌شده است.



شکل (۱۵): سازوکار ساخت و ارزیابی مدل طبقه‌بندی

همان‌طور که در شکل (۱۵) نشان داده‌شده است، داده‌های آموزشی که توسط زیرمجموعه انتخابی تعیین‌شده‌اند، به الگوریتم‌های طبقه‌بندی داده می‌شود. الگوریتم‌های طبقه‌بندی بر اساس داده‌های آموزش، مدل آموزشی را ایجاد می‌نمایند. بعد از ساخت مدل موردنظر، از داده‌های آزمون جهت ارزیابی مدل استفاده می‌گردد که نتایج این ارزیابی در ماتریس مجتمع‌سازی‌شده نگه‌داری می‌شود. درنهایت، توسط رأی‌گیری اکثریت نتیجه نهایی حاصل می‌گردد. در ادامه رابطه رأی‌گیری اکثریت نشان داده شده است.

$$\sum_{t=1}^{T} d_{t,j} = \max_{j=1}^{C} \sum_{t=1}^{T} d_{t,j} \qquad \text{(رابطه۶)}$$

در رابطهٔ (۶)، T نشان‌دهنده تعداد الگوریتم‌های طبقه‌بندی کننده است که برابر سه است. C نشان‌دهنده کلاس‌ها و $d_{t,j}$ نشان‌دهنده انتخاب کلاس jام توسط طبقه‌بندی tام است.

۵- ارزیابی روش پیشنهادی

در این بخش، ابتدا مشخصات محیط شبیه‌سازی از دیدگاه نرم‌افزاری و سخت‌افزاری بیان می‌شود. بعد از بیان ویژگی‌های محیط شبیه‌سازی، پارامترهای ارزیابی جهت ارزیابی نتایج روش پیشنهادی بیان خواهد شد. پارامترهای ارزیابی بکار رفته در این مقاله، جزء پارامترهای طبقه‌بندی است. این عامل‌ها شامل معیارهای دقت، صحت، فراخوان و معیار F است. مشخصات مهم مجموعه‌داده شامل تعداد نمونه، تعداد ویژگی و تعداد کلاس است که پس از بیان مشخصات مجموعه‌داده، نتایج روش پیشنهادی ارائه خواهد شد. مقایسه‌ای بین روش پیشنهادی و روش بیان‌شده در مقاله مرجع صورت‌گرفته است. هدف از این آزمون، نشان‌دادن برتری روش پیشنهادی نسبت به روش بیان‌شده در مقاله مرجع است. در ادامه جزئیات بیان‌شده در بالا به‌صورت مفصل شرح داده شده است.

۵-۱- محیط شبیه‌سازی

در این بخش مشخصات محیط شبیه‌سازی مورداستفاده در این مقاله از منظر سخت‌افزاری و نرم‌افزاری بیان می‌گردد. نرم‌افزارهای مورداستفاده در این تحقیق عبارت است از نرم‌افزار متلب نسخه 2016a و ویندوز 64 bit نسخه 10. همچنین مشخصات محیط سخت‌افزاری مورداستفاده نیز عبارت است از CPU: Core i7 8500 و RAM: 16 GB.

۵-۲- مجموعه‌داده

در این مقاله، جهت ارزیابی روش پیشنهادی از نسخه بهبودیافته مجموعه‌دادهٔ فونتانا[1] استفاده شده است. مجموعه‌داده اصلاح‌شده، حاصل ۷۴ نرم‌افزار متن‌باز است که حاوی ۵۷۰۰۰ رکورد و در پنج دسته گروه‌بندی شده است. از گروه‌بندی‌های موجود، چهار کلاس متعلق به بوی کد و یک کلاس مربوط به غیر بوی کد است. رده‌های مربوط به بوی کد شامل رده‌های Data Class، Feature Envy، God Class و Long Method است که تعداد نمونه‌های موجود در هر کلاس به ترتیب برابر ۳۵۰۰، ۵۰۰۰، ۸۰۰۰ و ۱۲۰۰۰ نمونه است. تعداد ویژگی‌های موجود در این مجموعه‌داده بعد از اصلاح داده به ۱۲۲ ویژگی رسیده است.

۵-۳- پارامترهای ارزیابی روش پیشنهادی

یک مدل طبقه‌بند (مدل یادگیری ماشین)، پس از یادگیری و آموزش، نیاز به ارزیابی دارد. این کار با اجرای مجموعه‌داده آزمون[2] بر روی طبقه‌بند انجام می‌پذیرد. برای ارزیابی یک طبقه‌بند و مقایسه عملکرد آن با دیگر طبقه‌بندها، به یک معیار کمّی و مشخص نیاز است. در ادبیات موضوع، معیارها و اندازه‌گیری‌های مختلفی برای این منظور ارائه شده است که هر یک از آن‌ها، از یک بعد خاص، طبقه‌بند را مورد ارزیابی قرار می‌دهند. به‌منظور ارزیابی نتایج روش پیشنهادی، از معیارهای آماری شامل دقت، صحت، فراخوان و معیار F استفاده شده است. جزئیات این روابط در ادامه آورده شده است.

[1] https://github.com/D-a-r-e-k/Code-Smells-Detection
[2] Test dataset

رابطه (۷)
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

رابطه (۸)
$$Precision = \frac{TP}{TP + FP}$$

رابطه (۹)
$$Recall = \frac{TP}{TP + FN}$$

رابطه (۱۰)
$$F\ Measure = \frac{2 * Precision * Recall}{Precision * Recall}$$

در روابط (۷) تا (۱۰)، به طور کلی، اگر یک نمونه مثبت باشد و طبقه‌بند نیز آن را به عنوان یک نمونه مثبت طبقه‌بندی کند، آن نمونه به عنوان مثبت صحیح (TP[1]) شمارش می‌شود. اما اگر طبقه‌بند، آن نمونه مثبت را به اشتباه به عنوان یک نمونه منفی طبقه‌بندی کند، آن نمونه به عنوان منفی کاذب (FN[2]) شمارش می‌شود. اگر یک نمونه منفی باشد و طبقه‌بند نیز آن را به عنوان یک نمونه منفی طبقه‌بندی کند، آن نمونه به عنوان منفی صحیح (TN[3]) محاسبه شده، اما اگر به اشتباه آن نمونه به عنوان مثبت طبقه‌بندی شود، به عنوان مثبت کاذب (FP[4]) شمارش می‌شود. معیار دقت (Accuracy) یک معیار پرتکرار و رایج برای ارزیابی کارایی مدل طبقه‌بند است و به‌عنوان نسبت بین نمونه‌های طبقه‌بندی‌شده صحیح به تعداد کل نمونه‌ها تعریف می‌شود. معیار صحت/درستی (Precision) به این اشاره دارد که اندازه‌گیری‌های یک آیتم چقدر به یکدیگر نزدیک هستند. این معیار، تعداد موارد مثبت طبقه‌بندی شده‌ای است که در واقعیت نیز موارد مثبتی هستند. معیار فراخوان (Recall) به‌عنوان درصد موارد بوی کد که روش پیشنهادی قادر است آن‌ها را به عنوان بوی کد پیش‌بینی کند، تعریف می‌شود. بنابراین هرچه مقدار فراخوانی بالاتر باشد، مطلوب‌تر و مناسب‌تر است. این معیار می‌تواند کامل بودن را اندازه‌گیری کند و در واقع نشان می‌دهد که چه مقدار از مواردی که واقعاً بوی کد هستند توسط روش پیشنهادی تشخیص داده شده‌اند. معیار امتیاز F (F-Measue) یک نوع میانگین از معیارهای صحت و فراخوانی است. امتیاز F، هر دو بخش FP و FN را در نظر گرفته و تعادل بین صحت و فراخوانی را حفظ می‌کند.

۵-۴- نتایج ارزیابی روش پیشنهادی

در این بخش، نتایج روش پیشنهادی به تفکیک رده‌های بوی کد در جدول (۳) آورده شده است که این نتایج بر اساس معیارهای ارزیابی بیان‌شده در بخش (۳-۳) است.

جدول (۳). نتایج روش پیشنهادی برروی مجموعه داده‌ی فونتانا

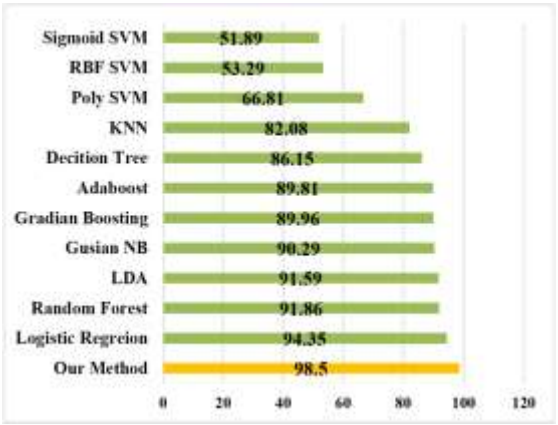
	Code Smell Type	ACC	REC	PRE	F-Measure
Our method	DataClass	98.50	96.25	97	96.60
	Feature Envay	93.5	92	92.5	92.25
	God Class	93	92.5	90	91.25
	Long Method	95	93.5	91.5	92.45

همان‌طور که در جدول (۳) نشان داده‌شده است، روش پیشنهادی به طور کلی در تشخیص بوهای کد، عملکرد مطلوبی داشته است. روش پیشنهادی در تشخیص بوی کد Data Class به دلیل پیچیده نبودن این بوی کد دارای عملکرد بهتری نسبت به دیگر بوهای کد است. ضعیف‌ترین عملکرد روش پیشنهادی مربوط به God Class است که برابر ۹۳ درصد است و نشان‌دهنده پیچیدگی این بوی کد است.

همان‌طور که در جدول (۳) مشاهده می‌شود، بهترین عملکرد الگوریتم روش پیشنهادی مربوط به بوی کد Data Class با دقت ٪۹۸،۵۰ است و کمترین مقدار دقت، مربوط به بوی کد God Class با ٪۹۳ است.

۵-۵- مقایسه با کارهای مرتبط قبلی

در این بخش، نتایج روش پیشنهادی بر روی مجموعه‌داده فونتانا با دیگر بیان‌شده در [۲۴]روی همان مجموعه‌داده آورده شده است. این روش‌ها شامل Sigmoid SVM، RBF SVM، KNN، Poly SVM، Decision Tree، Logistic Regresion، Gradian Boosting، Adaboost، Random Forest، Gusian NB و LDA می‌باشد. در شکل (۱۶) روش پیشنهادی با بیان‌شده در [۲۴]بر روی بوی کد Data Class مقایسه شده است.

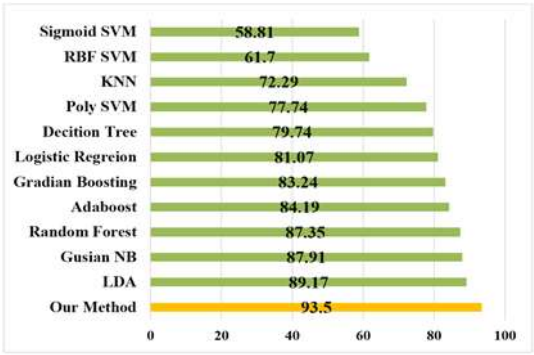


شکل (۱۶). مقایسه دقت روش پیشنهادی و [۲۴]در بخش Data

[1] True Positive
[2] False Negative
[3] True Negative
[4] False Positive

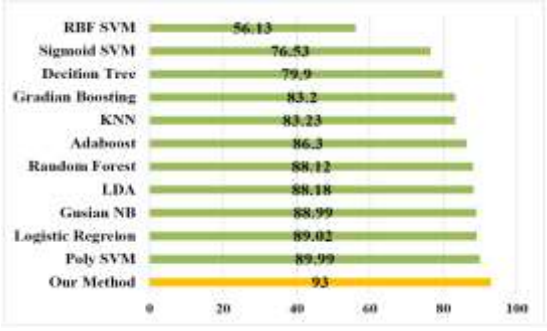
Class بر اساس معیار ACC

همان‌طور که در شکل (۱۶) مشاهده می‌شود، دقت روش پیشنهادی در مقایسه با مقاله [۲۴] بیشتر است. دلیل برتری روش پیشنهادی نسبت به [۲۴]مربوط به ترکیب دو بخش انتخاب ویژگی و ترکیب طبقه‌بندها است. در شکل (۱۶) بیشترین اختلاف مربوط به طبقه‌بند پشتیبان با ۵۱,۸۹٪ و کمترین اختلاف مربوط به طبقه‌بند لجستیک رگرسیون با ۹۴,۳۵٪ است. در شکل (۱۷) نتایج روش پیشنهادی بر روی بوی کد Feature Envy آورده شده است.



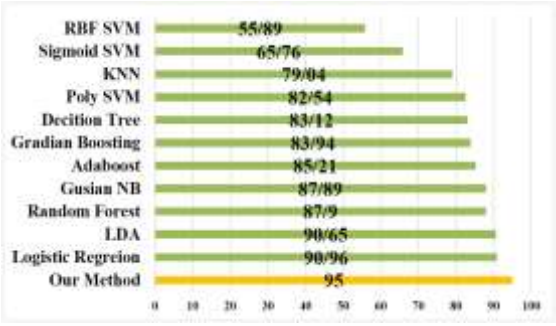
شکل (۱۷): مقایسه دقت روش پیشنهادی و [۲۴]در بخش Feature Envy بر اساس معیار ACC

همان‌طور که در شکل (۱۷) نشان داده شده است، عملکرد روش پیشنهادی در مقایسه با بیان‌شده در [۲۴]بهتر است. زیرا در روش پیشنهادی زیرمجموعه‌ای از ویژگی‌های پراهمیت استخراج می‌گردد که مهم‌ترین خاصیت این زیرمجموعه، افزایش کارایی الگوریتم طبقه‌بندی است. شکل (۱۷) نشان می‌دهد که روش پیشنهادی توانسته است توسط الگوریتم گرگ خاکستری ویژگی‌های مفید و تأثیرگذار را انتخاب نماید که این امر باعث کارایی روش پیشنهادی شده است. در شکل (۱۸) نتایج روش پیشنهادی بر روی God Class آورده شده است.



شکل (۱۸). مقایسه دقت روش پیشنهادی و [۲۴]در بخش God Class بر اساس معیار ACC

همان‌طور که در شکل (۱۸) نشان داده شده است، عملکرد روش پیشنهادی بر روی بوی کد God Class نیز همانند دو کلاس دیگر، بهتر است. در شکل (۱۹) نتایج روش پیشنهادی بر روی دقت شناسایی بوی کد Long Method آورده شده است.



شکل (۱۹): مقایسه دقت روش پیشنهادی و [۲۴]در بخش Long Method بر اساس معیار ACC

همان‌طور که در شکل (۱۹) نشان داده شده است، عملکرد روش پیشنهادی بروی Long Method نیز نسبت به بیان‌شده در [۲۴]بهتر است. کمترین اختلاف روش پیشنهادی نسبت به روش [۲۴]با الگوریتم طبقه‌بندی لجستیک برابر ۹۰,۹۶٪ و بیشترین اختلاف مربوط به الگوریتم طبقه‌بندی پشتیبان برابر ۵۵,۹۸٪ است.

در جمع‌بندی این بخش، دلیل برتری روش پیشنهادی به طور مختصر موردبحث و بررسی قرار می‌گیرد. همان‌طور که قبلاً نیز اشاره شد، بسیاری از تحقیقات انجام‌شده برای ساخت مدل‌های پیش‌بینی‌کننده بوهای کد، مبتنی بر الگوریتم‌های انتخاب ویژگی و یادگیری ماشین استوار است. از سوی دیگر، بسیاری از مطالعاتی که اخیراً در این حوزه انجام شده است، گویای این حقیقت است که فنون مجموعه‌ای[1]، برای پیش‌بینی بوی کد، دقت پیش‌بینی بالاتری نسبت به پیش‌بینی‌کننده‌های بوی کد مستقل[2] ایجاد کرده‌اند [۲۵]. ازاین‌رو، باتوجه‌به اینکه مدل طبقه‌بندی در روش پیشنهادی ما از ترکیب سه الگوریتم یادگیری SVM، KNN و DT تشکیل شده است، لذا عملکرد بهتری نسبت به عملکرد انفرادی هر یک از آنها ارائه کرده است. این امر می‌تواند یک فرصت روشن برای جامعه تحقیقاتی تلقی شود تا با رویکرد جدید نسبت به بهبود تشخیص ناهنجاری‌ها و مشکلات ساختاری برنامه‌های نرم‌افزاری، اقدام نمایند.

۶- نتیجه‌گیری و کارهای آینده

همان‌طور که در بخش‌های پیشین بیان شد، به زبان ساده، بوی کد نتیجه برنامه‌نویسی ضعیف یا نادرست است. به طور معمول، بوی کد ناشی از عدم نوشتن کد مطابق با استانداردهای لازم است. بوی کد معمولاً به مشکلات طراحی، عدم رعایت اصول و قواعد برنامه‌نویسی، کد تکراری، پیچیدگی زیاد، ناهماهنگی بین

[1] Ensemble Techniques
[2] Stand-alone Code Smell Predictors

قسمت‌های مختلف کد و موارد مشابه اشاره می‌کند. در صورت برطرف‌نکردن بوهای رمز، ممکن است منجر به چالش‌هایی از جمله کاهش قابلیت نگهداری، دشواری در توسعه و افزایش احتمال وجود خطاها در نرم‌افزار شود؛ بنابراین، شناسایی این نوع رمزها، نقشی تأثیرگذار و مهم در آزمون و تضمین کیفیت نرم‌افزار ایفا می‌کند. شناسایی بوهای کد، عموماً توسط فنون یادگیری ماشین انجام می‌شود. عملکرد این فنون بسیار وابسته به داده آموزشی است و داده آموزشی خوب منجر به ساخت یک مدل یادگیری قدرتمند می‌شود. یک مجموعه‌داده دارای ویژگی‌های نرم‌افزاری زیادی است و یکی از شاخص‌های یک داده آموزشی مناسب، نبود ویژگی‌های نامرتبط و افزونه در آن است. حذف این ویژگی‌ها با استفاده از انتخاب ویژگی انجام می‌شود که استفاده از این روش‌ها امری اجتناب‌ناپذیر به نظر می‌رسد.

یکی از بهبود عملکرد طبقه‌بند و مدل شناسایی بوی کد، انجام یک انتخاب ویژگی خوب و قدرتمند بر روی مجموعه‌داده است. برای فائق آمدن بر چالش‌های شناسایی بوی کد، از ی استفاده می‌شود که معمولاً دارای سه بخش کلی زیر است: ۱- جستجو به‌منظور یافتن ویژگی‌های بهینه، ۲- ساخت طبقه‌بند بر اساس ویژگی‌های بهینه و ۳- ارزیابی کارایی مدل طبقه‌بند. در این مقاله، از الگوریتم گرگ خاکستری جهت انتخاب ویژگی‌های مفید و تأثیرگذار استفاده شده است. این الگوریتم برخلاف سنتی انتخاب ویژگی، به‌صورت تکاملی عمل می‌کند و توانایی پیداکردن نقاط بهینه و جستجو در فضای مسئله را دارد. ازجمله مزایای این الگوریتم برای انتخاب ویژگی، می‌توان به کارایی بالاتر، قابلیت استفاده در مسائل پیچیده، سرعت بالا و قابلیت انعطاف‌پذیری آن اشاره کرد. در این مقاله، برای ساخت مدل طبقه‌بند یا مدل یادگیری ماشین، از ترکیب سه الگوریتم نزدیک‌ترین همسایه، درخت تصمیم و پشتیبان استفاده شده است که هر سه از قدرتمند حوزه خود محسوب می‌شوند. ترکیب این سه روش و استفاده از مزایای آنها به طور هم‌زمان، همراه با استفاده از قابلیت رأی‌گیری اکثریت، جزء نوآوری این مقاله بوده و یک روش قدرتمند برای ایجاد مدل طبقه‌بندی فراهم می‌کند. جهت ارزیابی روش پیشنهادی، از مجموعه‌داده فونتانا که شامل بوهای کد متد طولانی، خصیصه حسادت، کلاس خدا و کلاس داده است، استفاده شده است. نتایج روش پیشنهادی بر اساس معیار دقت (Acc)، برای تشخیص بوهای کد به ترتیب ۹۵، ۹۳,۵، ۹۳ و ۹۸,۵ درصد به‌دست‌آمده است که نسبت به سایر طبقه‌بندی از جمله؛ SVM با کرنل‌های مختلف، Random Forest، KNN و غیره عملکرد بهتری را نشان می‌دهد.

در خصوص کارهای آیند، برای بهبود انتخاب ویژگی که نقش کلیدی در بهبود عملکرد مدل‌های یادگیری و طبقه‌بندی ایفا می‌کنند، می‌توان پیشنهادهایی به شرح زیر ارائه نمود: ۱- استفاده از معیارهایی به جز Accuracy برای ارزیابی مدل طبقه‌بند. ۲- استفاده از ترکیب الگوریتم گرگ خاکستری با دیگر الگوریتم‌های تکاملی نظیر الگوریتم تبرید، ازدحام ذرات و غیره برای بهبود فرایند انتخاب ویژگی ۳- استفاده از فنون و متعادل‌سازی داده در مجموعه‌داده مثل نمونه‌برداری Resample.

۷- مراجع

[1] M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, D. Poshyvanyk, "When and Why Your Code Starts to Smell Bad," IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 1, pp. 403–414, 2017.

[2] M. S. Haque, J. Carver, T. Atkison, "Causes, impacts, and detection approaches of code smell: A survey," ACMSE '18: Proceedings of the ACMSE Conference, pp. 1-8, 2018.

[3] M. Fowler, "Refactoring: Improving the Design of Existing Code," Addison-Wesley Professional 2 Ed, pp. 40–48, 2018.

[4] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, and A. De Lucia, "On the diffuseness and the impact on maintainability of code smells: A large scale empirical investigation," in Proceedings of the 40th International Conference on Software Engineering (ICSE), pp. 1188–1221, ACM, 2018.

[5] A. Tahir, J. Dietrich, S. Counsell, S. Licorish, and A. Yamashita, "A large scale study on how developers discuss code smells and anti-pattern in stack exchange sites," Information and Software Technology, vol. 125, pp. 30–36, 2020.

[6] X. Han, A. Tahir, P. Liang, S. Counsell, Y. Luo, "Understanding Code Smell Detection via Code Review: A Study of the OpenStack Community," In: IEEE/ACM 29th International Conference on Program Comprehension (ICPC) [Internet]. IEEE, pp.323–34. 2021.

[7] M. Fowler, "Refactoring: Improving the Design of Existing Code," Addison-Wesley Professional, pp. 25–33 2018.

[8] G. Langelier, H. Sahraoui, P. Poulin, "Visualization-based analysis of quality for large-scale software systems," in: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, pp. 214–223, 2014.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, "The weka data mining software: an update," ACM SIGKDD Explor. Newsl, vol. 11 (1), pp. 10–18, 2015.

[10] R. Marinescu, "Detection strategies: metrics-based rules for detecting design flaws," in: 20th IEEE International Conference Proceedings on Software Maintenance, pp. 350–359. 2004.

[11] G. Ganea, I. Verebi, R. Marinescu, "Continuous quality assessment with incode," Sci. Comput. Program, vol 134, pp. 19–36, 2017.

[12] M. Mantyla, "Bad smells in software-a taxonomy and an empirical study," Helsinki University of Technology, pp. 303–314, 2003.

[13] B. Venkatesh, J. Anuradha, "A Review of Feature Selection and Its Methods," Cybern Inf Technol [Internet], vol 19(1), pp.3–26, 2019.

[14] S. Umadevi, KSJ. Marseline, "A survey on data mining classification algorithms," Proc IEEE Int Conf Signal Process Commun ICSPC, pp. 264–8, 2018.

[15] S. Kanj, F. Abdallah, "Editing training data for multi-label classification with the k-nearest neighbor rule,"

Pattern Analysis and Applications, Vol 19(1), pp. 145-161, 2016.
[16] S. Archana, and K. Elangovan, "Survey of classification techniques in data mining," International Journal of Computer Science and Mobile Applications, Vol 2(2): pp. 65-71. 2014.
[17] S. Roy, S. Mondal, A. Ekbal, MS. Desarkar, "Dispersion Ratio based Decision Tree Model for Classification," Expert Syst Appl [Internet], vol 116, pp. 1–9, 2019.
[18] S .Huang, CAI. Nianguang, P. Penzuti, S. Narandes, Y. Wang, XU. Wayne, "Applications of support vector machine (SVM) learning in cancer genomics," Cancer Genomics and Proteomics, vol 15(1), pp. 41–5, 2019.
[19] EO. Kiyak, D. Birant, KU. Birant, "Comparison of Multi-Label Classification Algorithms for Code Smell Detection," In: 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) [Internet], IEEE, pp.1–6, 2019.
[20] F. Pecorelli, D. Di Nucci, C. De Roover, A. De Lucia, "On the role of data balancing for machine learning-based code smell detection," MaLTeSQuE 2019 - Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, co-located with ESEC/FSE. pp. 19–24, 2019.
[21] F. Pecorelli, F. Palomba, D. Di Nucci, A. De Lucia, "Comparing Heuristic and Machine Learning Approaches for Metric-Based Code Smell Detection," In: IEEE/ACM 27th International Conference on Program Comprehension (ICPC) [Internet], pp. 93–104, 2019.
[22] R. Ibrahim, M. Ahmed, R. Nayak, S. Jamel, "Reducing redundancy of test cases generation using code smell detection and refactoring." J King Saud Univ - Comput Inf Sci [Internet], vol 32(3) , pp. 367–74, 2020.
[23] T. Guggulothu, SA. Moiz, "Code smell detection using multi-label classification approach," Software Quality Journal, Vol. 28, pp. 63–86 , 2020.
[24] S. Jain, A. Saha, "Improving performance with hybrid feature selection and ensemble machine learning techniques for code smell detection." Sci Comput Program [Internet]. Dec, vol 212, pp. 1–34, 2021.
[25] Muhammad Ilyas Azeem, Fabio Palomba, Lin Shi, Qing Wang, "Machine Learning Techniques for Code Smell Detection:
[26] A Systematic Literature Review and Meta-Analysis" Information & Software Technology. January 7, 2019.