مقاله ۱ : طبقه بندی چندبرچسبی برای پیش بینی نقص در مهندسی نرم‌افزار

مرجع:

Pachouly, J., Ahirrao, S., Kotecha, K., Kulkarni, A., & Alfarhood, S. (2025) Multilabel classification for defect prediction in software engineering. Scientific Reports, 15(1), 8739. [DOI: 10.1038/s41598-025-93242-8]

این مطالعه به مسئله پیش‌بینی نقص نرم‌افزار از دیدگاه طبقه‌بندی چندبرچسبی[۱] نگاه می‌کند و استدلال کرده است که در دنیای واقعی، یک نقص نرم‌افزاری می‌تواند همزمان متعلق به چندین دسته باشد؛ برای نمونه، یک نقص ممکن است هم یک «مشکل بحرانی» [۲] باشد و هم برای رفع آن به «بهبود کد» [۳] نیاز داشته باشد و هم مستلزم به روزرسانی مستندات [۴] باشد. روش کار بر روی یک مجموعه داده سفارشی ساخته شده از مخزن Spring Framework در گیت‌هاب انجام شده است [۱].

فرآیند پیش‌پردازش داده شامل ایجاد یک خلاصه جامع از گزارش نقص با استفاده از عنوان، بدنه، نظرات توسعه‌دهندگان و تسترها، و قطعات کد است. برای انتخاب ویژگی و کاهش ابعاد داده از آزمون کای‌اسکوئر [۵] و برای مقابله با مشکل شدید عدم تعادل کلاس‌ها در داده‌های چندبرچسبی از روش Non-Negative Least Squares (NNLS) استفاده شده است. برای آموزش مدل‌ها، از روش‌های سنتی یادگیری ماشین [۶] شامل نیو بیز،[۷] رگرسیون لجستیک [۸] و جنگل تصادفی [۹] و همچنین از روش‌های یادگیری عمیق [۱۰] شامل پرسپترون چندلایه [۱۱] شبکه‌های عصبی کانولوشنی [۱۲] همراه زنجیره‌های طبقه بندی کننده [۱۳] به منظور حفظ وابستگی‌های بین برچسب‌ها بهره گرفته شده است [۱].

---

[1] Multilabel Classification
[2] Blocker
[3] Enhancement
[4] Documentation
[5] Chi-square
[6] Machine Learning (ML)
[7] Multinomial Naive Bayes
[8] Logistic Regression
[9] Random Forest
[10] Deep learning (DL)
[11] MLP
[12] CNN
[13] Classifier Chains

نتایج تجربی نشان می‌دهند که متوازن‌سازی مجموعه داده با استفاده از روش *NNLS* تأثیر چشمگیری در بهبود عملکرد مدل‌ها داشته است. بهترین عملکرد مربوط به مدل *CNN* بر روی داده‌های متوازن شده با مقادیر فراخوانی[1] برابر با ۹۴/۸۴٪، *F1-Score* برابر با ۹۴/۶۰٪ و *Hamming Loss* برابر با ۰/۰۵۱۹ گزارش شده است. این نتایج نشان‌دهنده برتری چشمگیر رویکرد پیشنهادی نسبت به بسیاری از روش‌های پیشین در حوزه پیش‌بینی نقص چندبرچسبی بوده است [۱].

نقاط قوت این مقاله شامل نوآوری اصلی این پژوهش است که در رویکرد چندبرچسبی برای مدل‌سازی پیچیدگی نقص‌های نرم‌افزاری بوده به طوری که به واقعیت دنیای صنعت نزدیک‌تر بوده است. ارائه یک چارچوب جامع برای بازنمایی نقص که اطلاعات مختلف را ادغام می‌کند و قابلیت تفسیرپذیری بالایی دارد. همچنین، معرفی روش NNLS برای متوازن‌سازی داده‌های چندبرچسبی، یک راهکار مؤثر برای غلبه بر چالش دیرینه عدم تعادل کلاس‌ها محسوب می‌شود .

نقاط ضعف این مقاله هم به مجموعه داده مورد استفاده برمیگردد که در این مطالعه به یک مخزن خاص [2] محدود بوده و ممکن است نتایج برای پروژه‌های با زبان‌های برنامه‌نویسی و ساختارهای متفاوت، از تعمیم‌پذیری کافی برخوردار نباشد. همچنین، فرآیند جمع‌آوری داده‌ها به صورت نیمه‌خودکار انجام شده که ممکن است به کیفیت داده‌های ورودی وابسته باشد و خطاهای انسانی در آن تأثیرگذار باشد .

مقاله ۲: معماری‌های یادگیری عمیق برای پیش‌بینی نقص نرم‌افزار با در نظر گرفتن تأثیر متریک‌های خطا و عدم تعادل کلاس

مرجع:

Althiban, A. S., Alharbi, H. M., Al Khuzayem, L. A., & Eassa, F. E. (2025). Automated Models for Predicting Software Defects in Hybrid MPI and OpenMP Parallel Programs Using Deep Learning. IEEE Access, 13, 65939-65964. [DOI: 10.1109/ACCESS.2025.3559807]

[1] Recall
[2] Spring Framework

مقاله [۲] به ارائه و مقایسه مدل‌های یادگیری عمیق برای پیش‌بینی نقص‌های همگام‌سازی [۱]مانند بن‌بست[۲] و شرط رقابت [۳]در برنامه‌های موازی ترکیبی MPI و OpenMP می‌پردازد. روش کار شامل استخراج توکن‌های کد بر اساس درخت نحو انتزاعی [۴]با استفاده از کتابخانه Clang و همچنین روش‌های مبتنی بر عبارت منظم[۵] است. سه معماری شبکه عصبی شامل شبکه عصبی کانولوشنی ، شبکه حافظه کوتاه‌مدت طولانی[۶]و مدل ترکیبی CNN-LSTM بر روی یک مجموعه داده متوازن شامل ۱۵۰۰ فایل *C++* ارزیابی شده‌اند. این مطالعه به تأثیر روش‌های مختلف توکن‌سازی و نقش آن در استخراج ویژگی‌های ساختاری و معنایی کد برای پیش‌بینی نقص پرداخته است [۲].

نتایج بدست آمده نشان می‌دهند که نمایش مبتنی بر توکن‌های استخراج شده توسط *Clang* مؤثرترین ورودی برای مدل های مبتنی بر الگوریتم یادگیری عمیق بوده و مدل *CNN* با دقت ۹۷٪ و منحنی زیر نمودار [۷]معادل ۹۹٪ بهترین عملکرد را در بین مدلهای مورد بررسی داشته است. مدل *CNN* به دلیل توانایی در استخراج روابط ساختاری در کد توکن‌شده، از مدلهای *LSTM* و *CNN-LSTM* که به ترتیب اجرا [۸]حساس هستند، عملکرد بهتری از خود نشان داده است. این یافته نشان می‌دهد که برای زبان‌هایی با ساختار صریح مانند *C++*، استخراج ویژگی‌های ساختاری از اهمیت بیشتری نسبت به ترتیب اجرا برخوردار است [۲].

این مطالعه از اولین پژوهش‌هایی است که به طور خاص به کاربرد یادگیری عمیق برای پیش‌بینی نقص در برنامه‌های موازی ترکیبی پرداخته و چالشی مهم در حوزه محاسبات با کارایی بالا [۹]را هدف قرار داده است. حجم مجموعه داده نسبت به پژوهش‌های پیشین به طور قابل توجهی افزایش یافته و از تنوع بیشتری برخوردار است. همچنین، مقایسه سیستماتیک سه معماری مختلف یادگیری عمیق، بینش ارزشمندی در مورد انتخاب مدل مناسب برای این حوزه ارائه می‌دهد [۲]. از سوی دیگر، رویکرد توکن‌سازی استفاده شده در این مطالعه، ترتیب اجرا را به صراحت حفظ نکرده که همین موضوع ممکن است بر عملکرد مدل‌های توالی‌محور مانند LSTM تأثیر منفی گذاشته باشد. همچنین، مجموعه داده های بکار برده شده صرفا بر روی نقص‌های بن‌بست و شرط مسابقه متمرکز بوده است و تعمیم‌پذیری مدل به سایر انواع نقص مانند نشت حافظه را با محدودیت مواجه می‌سازد.

---

1 Synchronization Defects
2 Deadlock
3 Race Condition
4 AST
5 Regex
6 LSTM
7 AUC
8 Sequence
9 HPC

در نهایت، یافته‌های این مطالعه به برنامه‌های *C++* با *MPI/OpenMP* محدود بوده و برای سایر پارادایم‌های موازی قابل تعمیم نیست .

مقاله ۳: چارچوب MSTDP برای پیش‌بینی نقص در لحظه با استفاده از ویژگی‌های زمانی چندمقیاسه و توجه متقابل

مرجع:

Liu, Z., Yuan, C., Liu, H., Li, X., Liu, S., Zhou, X., & Tian, Z. (2026). MSTDP: a multi-scale temporal deep learning framework for just-in-time software defect prediction with cross-attention fusion. Journal of King Saud University - Computer and Information Sciences, 38(2), 101634. [DOI: 10.1007/s44443-025-00401-y]

در [۳]، یک چارچوب یادگیری عمیق مبتنی بر زمان چندمقیاسه به نام *MSTDP* برای پیش‌بینی نقص در لحظه [1]ارائه می‌دهد. این روش با هدف غلبه بر محدودیت روش‌های سنتی که تغییرات کد را به عنوان رویدادهای ایزوله و ایستا در نظر می‌گیرند، طراحی شده است. چارچوب *MSTDP* با استخراج ویژگی‌های زمانی در سه مقیاس مختلف (ساعت، روز، هفته) و ترکیب آن‌ها با اطلاعات معنایی کد، رفتار توسعه‌دهنده و آگاهی از چرخه حیات پروژه، سعی در مدل‌سازی پویای فرآیند توسعه دارد. برای ترکیب عمیق این ویژگی‌های ناهمگن از مکانیزم توجه متقابل [2]و برای افزایش مقاومت مدل در برابر تغییرات مفهومی [3]در طول زمان، یک ماژول تحلیل تغییرات مفهومی طراحی شده است.

ارزیابی بر روی ۹ پروژه بزرگ متن‌باز نشان داده است که روش *MSTDP* در تمامی معیارهای ارزیابی شامل دقت[4](۷۶/۸۹٪)، صحت [5](۶۱/۰۲٪)، فراخوانی (۶۷/۸۲ ٪)، *F1-Score* (۶۰/۹۳ ٪) و *AUC* برابر با (۸۱/۸۸ ٪) عملکرد بهتری نسبت به ۷ روش برتر پایه مانند *JITBoost* و *FENSE* داشته است. آزمون آماری *Wilcoxon signed-rank* نیز معناداری این بهبودها را تأیید کرده است. به ویژه، مدل در برابر تغییرات مفهومی از مقاومت و تعمیم‌پذیری بیشتری برخوردار بوده که نشان‌دهنده برتری آن در محیط‌های پویای توسعه نرم‌افزار است [۳].

[1] Just-In-Time
[2] Cross-Attention
[3] Concept Drift
[4] Accuracy
[5] Precision

بدین ترتیب از نقاط قوت [۳]، رویکرد نوآورانه این تحقیق در استفاده از ویژگی‌های زمانی چندمقیاسه بوده است که تصویر کاملتری از پویایی‌های توسعه ارائه میدهد. همچنین، پرداختن به موضوع مهم تغییرات مفهومی [۱] که در بسیاری از پژوهش‌ها نادیده گرفته می‌شود و ارائه راهکاری عملی برای افزایش مقاومت مدل در برابر آن، از دیگر نقاط قوت این مقاله محسوب می‌شود. همچنین، استفاده از مکانیزم توجه متقابل برای ترکیب بهینه ویژگی‌های مختلف، رویکردی نوین و کارآمد است [۳].

افزایش پیچیدگی محاسباتی و نیاز به داده‌های تاریخی بیشتر برای استخراج ویژگی‌های زمانی نیز، از جمله محدودیت‌ها و نقاط ضعف این روش بوده است که ممکن است پیاده‌سازی آن را برای پروژه‌های کوچک و با دوره توسعه کوتاه، با چالش مواجه کند. همچنین، پیاده‌سازی دقیق این چارچوب نیازمند تنظیم دقیق پارامترهای متعدد است که ممکن است فرآیند آموزش مدل را دشوار سازد [۳].

مقاله ۴: افزایش دقت پیش‌بینی نقص نرم‌افزار از طریق هوش هیبریدی مبتنی بر انتخاب ویژگی

مرجع:

Kumar, V. K., Maram, B., & Chandre, S. (2025). Enhancing Software Defect Prediction Accuracy through Feature Selection-Driven Hybrid Intelligence. In 2025 Third International Conference on Industry 4.0 Technology (I4Tech) (pp. 1-5). IEEE. [DOI: 10.1109/I4Tech64670.2025.11277790]

مقاله [۴]، یک مدل هوش هیبریدی ترکیبی از دو الگوریتم جنگل تصادفی و ماشین بردار پشتیبان [۲] ارائه می‌دهد که با استفاده از فرآیند انتخاب ویژگی، دقت پیش‌بینی نقص را بهبود می‌بخشد. برای انتخاب ویژگی‌ها، از دو روش *RFE* و[۳] *Mutual Information* بر روی مجموعه داده‌های عمومی *PROMISE* و *NASA MDP* استفاده شده است. هدف از انتخاب ویژگی، کاهش پیچیدگی مدل و حذف ویژگی‌های زائد و کم اهمیت است تا مدل بتواند با داده‌های با کیفیت‌تر و ابعاد کمتر، عملکرد بهتری داشته باشد [۴].

[1] Concept Drift
[2] Support Vector Machine (SVM)
[3] Recursive Feature Elimination (RFF)

مدل هیبریدی پیشنهادی در [۴]، به دقت متوسط ۹۲/۴٪ دست یافته که نسبت به مدل‌های سنتی ۱۱/۳٪ بهبود را نشان می‌دهد. روش‌های انتخاب ویژگی منجر به کاهش ۲۵ درصدی پیچیدگی مدل و افزایش ۱۴ درصدی دقت شدند. آزمایش‌های تعمیم‌پذیری بر روی مجموعه داده‌های مختلف، موفقیت‌آمیز گزارش شده و نشان‌دهنده قابلیت اطمینان مدل در شناسایی ماژول‌های مستعد نقص است. ترکیب مؤثر دو الگوریتم قدرتمند یادگیری ماشین با روش‌های انتخاب ویژگی که منجر به مدلی با دقت بالا و پیچیدگی کمتر شده است، نقطه قوت اصلی این پژوهش محسوب می‌شود. استفاده از مجموعه داده‌های استاندارد و شناخته شده *PROMISE* و *NASA MDP* هم امکان مقایسه مستقیم با سایر پژوهشها را فراهم کرده است. علاوه بر این، کاهش چشمگیر پیچیدگی مدل، یک مزیت عملی برای پیاده‌سازی در سیستم‌های واقعی با منابع محدود محسوب می‌شود [۴].

از سوی دیگر، اطلاعات دقیقی در مورد نوع نقص‌های پیش‌بینی شده (نقص در سطح ماژول یا فایل) و معیارهای ارزیابی دیگر مانند فراخوانی و *F1-Score* در مقاله ارائه نشده است و صرفا به معیار دقت اکتفا شده بوده است. تکیه بر داده‌های NASA که قدمت زیادی دارند و مربوط به پروژه‌های قدیمی هستند، ممکن است کارایی مدل را برای پروژه‌های مدرن با معماری‌های جدید و زبان‌های برنامه‌نویسی متفاوت، محدود کند [۴].

مقاله ۵: پیش‌بینی نقص نرم‌افزار مبتنی بر ریسک با استفاده از متریک‌های خطا

مرجع:

Domain-specific implications of error-type metrics in risk-based software fault prediction

این مطالعه یک رویکرد مبتنی بر ریسک برای پیش‌بینی نقص ارائه داده است که ماژول‌ها را به سه دسته ریسک (بالا، متوسط، پایین) طبقه‌بندی می‌کند و از متریک‌های خطای ران‌تایم[1] در کنار متریک‌های سنتی استفاده می‌کند. برای تحلیل وابستگی‌های بین متغیرها از تحلیل مؤلفه‌های اصلی[2] و برای متوازن‌سازی داده‌ها از *SMOTE* استفاده شده است. الگوریتم‌های مورد استفاده شامل *SVM*، *Random Forest* و *XGBoost* هستند و بر روی داده‌های چهار پروژه متن‌باز اعتبارسنجی شده‌اند.

[1] Error-type Metrics
[2] PCA

مدل *XGBoost* بهترین عملکرد را با دقت ۹۷/۴٪، ضریب همبستگی متیوز [1]برابر با ۹۶/۱٪ و *F1-Score* برابر با ۹۷/۴٪ در سراسر مجموعه داده‌ها از خود نشان داد. تحلیل *PCA* نیز نشان داده است که تأثیر متریک‌های خطا بر روی ماژول‌های نرم‌افزاری وابسته به حوزه [2]است و عملکرد آن‌ها در پروژه‌های مختلف یکسان نیست.

از جمله نقاط قوت مقاله [۵]، ارائه یک چارچوب طبقه‌بندی ریسک که برای مدیریت پروژه و تخصیص منابع بسیار مفید بوده است به طوری که اطلاعات دقیق‌تری نسبت به مدل‌های دودویی در اختیار مدیران پروژه قرار می‌دهد. اعتبارسنجی قوی بر روی پروژه‌های مختلف که نشان‌دهنده تعمیم‌پذیری نسبی روش است.

از نقاط ضعف این تحقیق نیز می توان به این نکته اشاره کرد که نیازمند داده‌های دقیق از تعداد نقص‌ها در هر ماژول برای طبقه‌بندی ریسک بوده است که ممکن است در همه مخازن به راحتی در دسترس نباشد. همچنین، عملکرد بسیار بالای مدل *XGBoost* ممکن است نشان‌دهنده بیش برازش داده‌ها [3]باشد، هرچند اعتبارسنجی روی پروژه‌های متعدد تا حدی این نگرانی را کاهش داده است.

مقاله ۶: مرور سیستماتیک بر رویکردهای یادگیری ماشین برای پیش‌بینی نقص نرم‌افزار

مرجع:

Machine Learning Approaches for Software Defect Prediction

مطالعه انجام شده توسط پژوهشگران در [۶]، یک روش جدید ارائه نداده، بلکه محققان به تحلیل، طبقه‌بندی و ارزیابی پژوهش‌های موجود در این حوزه پرداخته اند. همچنین، چالش‌های اساسی مانند مهندسی ویژگی دستی، عدم تفسیرپذیری مدل‌های پیچیده، مشکل عدم تعادل کلاس‌ها و کمبود معیارهای ارزیابی استاندارد و یکپارچه در [۶] به صورت جامع بررسی شده اند. نتایج این مطالعه مروری نشان داده است که اگرچه الگوریتم‌های بهینه‌سازی شده می‌توانند به دقت بالایی دست پیدا کنند اما بسیاری از مدل‌ها بر روی مجموعه داده‌های محدود و قدیمی (مانند داده‌های *NASA*) ارزیابی شده‌اند و فاقد اعتبارسنجی کافی در دنیای واقعی هستند. چالش‌های اصلی شامل کیفیت پایین داده، عدم تعادل شدید کلاس‌ها و تفسیرپذیری پایین مدل‌های یادگیری عمیق به عنوان موانع کلیدی برای کاربرد عملی این روش‌ها شناسایی شده‌اند.

[1] MCC
[2] Domain-specific
[3] Overfitting

این مقاله یک دیدگاه جامع و به روز از وضعیت پژوهش‌ها و شناسایی شکاف‌های تحقیقاتی و چالش‌های عملی ارائه داده است که به محققان کمک می‌کند تا مسیرهای آینده را بهتر بشناسند. این مقاله می‌تواند به عنوان یک نقشه راه برای محققان تازه‌وارد به این حوزه عمل کند. از طرف دیگر، به عنوان یک مقاله مروری، فاقد روش یا نتایج تجربی جدید است و ارزش آن در جمع‌بندی و تحلیل پژوهش‌های پیشین خلاصه می‌شود. بنابراین، نمی‌توان از آن به عنوان منبعی برای بهبود مدل‌های موجود استفاده کرد.

مقاله ۷: پیش‌بینی کیفیت نرم‌افزار بر اساس شدت نقص

مرجع: مقاله منتشر شده در *Journal of Internet Computing and Services (KCI)*، سال ۲۰۱۵. این مقاله به دلیل قدمت بیشتر، روش‌های جدیدتری را پوشش نمی‌دهد اما به عنوان یک مطالعه پایه‌ای در حوزه طبقه‌بندی شدت نقص قابل توجه است.

این مطالعه به جای پیش‌بینی دودویی (نقص دارد/ندارد)، یک مدل طبقه‌بندی سه‌تایی [1]برای پیش‌بینی شدت نقص [2]ارائه می‌دهد. ورودی مدل، متریک‌های سنتی مانند اندازه و پیچیدگی کد است و از چهار الگوریتم یادگیری ماشین از جمله شبکه عصبی پس‌انتشار [3]برای آموزش استفاده شده است.

ارزیابی بر روی دو مجموعه داده *NASA* نشان داده است که مدل شبکه عصبی پس‌انتشار با دقت حدود ۸۱٪ و ۸۸٪ به ترتیب برای دو مجموعه داده، بهترین عملکرد را در بین سایر الگوریتم‌ها دارد. حرکت فراتر از پیش‌بینی دودویی و ارائه اطلاعات عملی‌تر برای فرآیند تست و تخصیص منابع (شناسایی ماژول‌های با نقص با شدت بالا). استفاده از داده‌های استاندارد *NASA* که امکان مقایسه با بسیاری از مطالعات دیگر را فراهم می‌کند، از جمله نقاط قوت مقاله [۷] بوده و از سوی دیگر هم، بکارگیری داده‌های قدیمی *NASA* که ممکن است با پروژه‌های مدرن تطابق نداشته باشد و همچنین مدل پیشنهادی تنها از متریک‌های سنتی استفاده کرده و از روش‌های جدیدتر مانند متریک‌های خطا یا یادگیری عمیق بهره‌ای نبرده است، به عنوان نقاط ضعف آن محسوب میشوند.

[1] Ternary Classification
[2] Severity
[3] Backpropagation Neural Network

مقاله ۸: کاربردهای صنعتی هوش مصنوعی در پیش‌بینی نقص نرم‌افزار

مرجع:

Industrial applications of artificial intelligence in software defects prediction

مقاله [۸] که در سال ۲۰۲۵ در مجله *Computers & Industrial Engineering*، منتشر شده است، بر اساس دستورالعمل *PRISMA*، ۴۵ مقاله از سال ۲۰۱۹ تا ۲۰۲۳ را از پایگاه‌های معتبر علمی تحلیل کرده است. این مطالعه یک مرور سیستماتیک برای بررسی کاربردهای هوش مصنوعی در پیش‌بینی نقص نرم‌افزار در محیط‌های صنعتی است و هدف آن شناسایی روندها، تکنیک‌ها، معیارها، ابزارها و مجموعه داده‌های پرکاربرد در این حوزه بوده تا شکاف بین تحقیقات دانشگاهی و نیازهای صنعت را مشخص کند [۸].

مهم‌ترین یافته‌ها نشان می‌دهند که الگوریتم‌های *SVM* (با ۴۲/۲۲٪) و *Random Forest* (با ۳۵/۵۶٪) پرکاربردترین تکنیک‌ها در صنعت هستند. معیارهای *Accuracy* و *F1-Score* (با ۶۸/۸۹٪) رایج‌ترین معیارهای ارزیابی هستند. همچنین، مخزن داده *NASA MDP* با ۳۱/۱۱٪ بیشترین استفاده را داشته و زبان پایتون با ۳۵/۵۶٪ محبوب‌ترین زبان برنامه‌نویسی برای پیاده‌سازی این سیستم‌هاست.

نقطه قوت مقاله [۸]، ارائه یک تصویر آماری و دقیق از روندهای تحقیقاتی در سال‌های اخیر و شناسایی روش‌ها و ابزارهای محبوب در صنعت است. این مطالعه منبعی معتبر برای انتخاب روش‌ها و ابزارهای مناسب برای پیاده‌سازی سیستم‌های پیش‌بینی نقص در محیط‌های صنعتی واقعی است؛ اما به عنوان یک مطالعه مروری، حاوی اطلاعات جدید و روش‌های نوآورانه نیست.

مقاله ۹ : پیش‌بینی نقص نرم افزار مبتنی بر شبکه‌های عمیق با رویکرد مقابله با عدم تعادل کلاس

مرجع:

Effective software defect prediction with deep neural networks

مقاله [۹]، یک مدل پیش‌بینی نقص مبتنی بر شبکه عمیق به نام *DN_SDP* ارائه داده است که به صورت کمی به بررسی تأثیر دو چالش اساسی در پیش‌بینی نقص یعنی حجم داده[۱] و عدم تعادل کلاس[۲] بر عملکرد مدل‌های یادگیری عمیق در مقایسه با مدل‌های سنتی یادگیری ماشین[۳] می‌پردازد. روش کار شامل اجرای آزمایش‌های گسترده بر روی ۱۱ مجموعه داده از مخزن استاندارد *PROMISE* و ارزیابی با ۸ معیار سنجش عملکرد است. به منظور مقابله با مشکل عدم تعادل کلاس‌ها، یک نوع بهبودیافته از مدل پایه به نام GAN به همراه *DN_SDP* ارائه شده است که از شبکه‌های مولد متخاصم[۴] برای تولید داده‌های مصنوعی مربوط به کلاس اقلیت (ماژول‌های معیوب) استفاده می‌کند و سپس این داده‌های متوازن شده به مدل DN_SDP داده می‌شوند.

اجرای مجموعه‌ای از آزمایش‌ها و آزمون آماری فریدمن با آزمون تعقیبی بونفرونی[۵] نشان داده است که مدل *DN_SDP* پیشنهادی در [۹]، به طور معنی‌داری مشکلات حجم داده و عدم تعادل کلاس را که در مدل‌های سنتی (*SN_SDP*) وجود دارد، کاهش می‌دهد. این مدل موفق به بهبود معیار *AUC* به میزان ٪۸، *F-measure* به میزان ٪۱۲ و MCC به میزان ٪۴۲ نسبت به مدل‌های سنتی شده است. مدل ترکیبی ارائه شده رد این مقاله نیز بهبودهای بیشتری را به همراه داشته و به ترتیب بهبود ٪۱۰، ٪۱۴ و ٪۲۵ در معیارهای *AUC* ، *F1-measure* و *MCC* نسبت به *DN_SDP* پایه نشان داده است. در مجموع، مدل *GAN_DN_SDP* از سایر مدل‌های پایه بهتر عمل کرده و افزایش ۱۷/۹ درصد در معیار *F-measure* را ثبت کرده است .

این مطالعه یکی از پژوهش‌های جامعی است که به طور همزمان تأثیر دو چالش مهم حجم داده و عدم تعادل کلاس را بر روی تعداد زیادی مجموعه داده (۱۱ مجموعه) بررسی کرده و از آزمون‌های آماری معتبر برای اعتبارسنجی نتایج استفاده نموده است. استفاده از *GAN* برای تولید داده‌های مصنوعی و مقایسه سیستماتیک عملکرد یادگیری عمیق در مقابل یادگیری سنتی، از نقاط قوت برجسته این مقاله محسوب میشود.

اگرچه در این مقاله از *GAN* برای متوازن‌سازی استفاده شده، اما همچنان چالش‌های مربوط به تولید داده‌های مصنوعی با کیفیت و واقع‌گرا در مجموعه داده‌های بسیار نامتوازن وجود دارد. همچنین، تمام مجموعه داده‌های مورد استفاده از مخزن *PROMISE* هستند که ممکن است تنوع کافی برای تعمیم‌پذیری کامل به پروژه‌های صنعتی با ساختارهای متفاوت را نداشته باشند.

---

[1] Dataset Size
[2] Class Imbalance
[3] Shallow Networks
[4] Generative Adversarial Network
[5] Bonferroni post-hoc Test

مقاله ۱۰ : پیش‌بینی نقص در سطح تیکت با رویکرد مجاورت زمانی

مرجع:

Anticipating bugs: Ticket-level bug prediction and temporal proximity effects

پژوهش [۱۰]، یک رویکرد نوین به نام پیش‌بینی در سطح تیکت (*Ticket-Level Prediction - TLP*) را معرفی کرده است که هدف آن پیش‌بینی این موضوع است که آیا یک تیکت (وظیفه) پس از پیاده‌سازی، منجر به ایجاد نقص در نرم‌افزار خواهد شد یا خیر. این روش در سه نقطه زمانی مختلف در چرخه حیات تیکت ارزیابی می‌شود: لحظه ایجاد[1] ، لحظه انتساب به توسعه‌دهنده[2] و لحظه پس از آخرین کامیت[3] مدل پیشنهادی از ۷۲ ویژگی متعلق به هفت خانواده مختلف شامل ویژگی‌های کد، توسعه‌دهنده، معیارهای *JIT* و ویژگی‌های مبتنی بر دما (تعداد کامیت‌ها و بازه‌های زمانی) استفاده می‌کند. ارزیابی بر روی حدود ۱۰،۰۰۰ تیکت از دو پروژه متن‌باز آپاچی با رویکرد پنجره‌لغزنده و سه طبقه‌بندی‌کننده یادگیری ماشین انجام شده است .

نتایج بدست آمده در این مقاله نشان می‌دهند که دقت پیش‌بینی *TLP* با نزدیک‌تر شدن به مرحله پیاده‌سازی (نزدیکی زمانی) افزایش می‌یابد که یک مقایسه ذاتی بین پیش‌بینی زودهنگام و دقت را تأیید می‌کند. به عبارت دیگر، هر چه تیکت به مرحله بسته شدن نزدیک‌تر باشد، اطلاعات بیشتری در دسترس است و پیش‌بینی دقیق‌تر می‌شود. از نظر قدرت پیش‌بینی هر خانواده از ویژگی‌ها، هیچ خانواده واحدی در تمام مراحل تسلط ندارد. در مراحل اولیه، ویژگی‌های مرتبط با توسعه‌دهنده بیشترین اطلاعات را دارند، در حالی که در مراحل نزدیک به بسته شدن، معیارهای کد و *JIT* اهمیت بیشتری پیدا می‌کنند. همچنین، ویژگی‌های مبتنی بر دما در تمام مراحل ارزش مکملی ارائه می‌دهند. نوآوری اصلی این مقاله در حرکت به سمت پیش‌بینی نقص در مرحله قبل از نوشتن کد (سطح تیکت) بوده است که با اصل «پیشگیری بهتر از درمان» همخوانی دارد و فرصت‌های جدیدی برای تخصیص ریسک‌آگاهانه تیکت‌ها و انتساب توسعه‌دهندگان فراهم می‌کند. تحلیل جامع ۷۲ ویژگی از خانواده‌های مختلف و بررسی تغییر قدرت پیش‌بینی آن‌ها در طول زمان، دیدگاه عمیقی نسبت به عوامل مؤثر بر نقص ارائه می‌دهد. نقاط ضعفش هم این است که رویکرد *TLP* نیازمند داده‌های تاریخی دقیق از چرخه حیات تیکت‌ها است که ممکن است در بسیاری از مخازن نرم‌افزاری به‌خوبی ثبت نشده باشد. همچنین، نتایج این مطالعه بر روی دو پروژه *apache* به دست آمده و ممکن است برای پروژه‌هایی با فرآیندهای توسعه متفاوت، قابل تعمیم نباشد.

---

[1] Open
[2] In Progress
[3] Closed

منابع

[1] Hong, E. S. (2015). *Software quality prediction based on defect severity. Journal of Internet Computing and Services*, 20(5), 73–81.

[2] Zaidi, H. Z., Ullah, U., Arshad, M., Aljuaid, H., Rauf, M. A., Sarwar, N., & Sajid, R. (2025). *Machine learning approaches for software defect prediction*. Applied Computational Intelligence and Soft Computing, 2025(1).

[3] Jing, X. (2023). Intelligent software defect prediction. Springer.

[4] Pachouly, J., Ahirrao, S., Kotecha, K., Kulkarni, A., & Alfarhood, S. (2025). *Multilabel classification for defect prediction in software engineering*. Scientific Reports, 15(1), Article 8739. https://doi.org/10.1038/s41598-025-93242-8

[5] Althiban, A. S., Alharbi, H. M., Al Khuzayem, L. A., & Eassa, F. E. (2025). *Automated models for predicting software defects in hybrid MPI and OpenMP parallel programs using deep learning*. IEEE Access, 13, 65939–65964. https://doi.org/10.1109/ACCESS.2025.3559807

[6] Liu, Z., Yuan, C., Liu, H., Li, X., Liu, S., Zhou, X., & Tian, Z. (2026). *MSTDP: A multi-scale temporal deep learning framework for just-in-time software defect prediction with cross-attention fusion*. Journal of King Saud University - Computer and Information Sciences, 38(2), Article 101634. https://doi.org/10.1007/s44443-025-00401-y

[7] Kumar, V. K., Maram, B., & Chandre, S. (2025). *Enhancing software defect prediction accuracy through feature selection-driven hybrid intelligence*. In 2025 Third International Conference on Industry 4.0 Technology (I4Tech) (pp. 1–5). IEEE. https://doi.org/10.1109/I4Tech64670.2025.11277790

[8] *Domain-specific implications of error-type metrics in risk-based software fault prediction*. Software Quality Journal, 33(7). (2025).

[9] Qiu, F., Yin, G., & Sun, Z. (2025). *Feature selection using a multi-strategy improved parrot optimization algorithm in software defect prediction*. PeerJ Computer Science, 11, e2815. https://doi.org/10.7717/peerj-cs.2815

[10] Falessi, D., et al. (2026). *Anticipating bugs: Ticket-level bug prediction and temporal proximity effects*. Empirical Software Engineering, 31, Article 43. https://doi.org/10.1007/s10664-025-10771-6