

# Detection of code smells using artificial intelligence: a systematic mapping study

*Detección de olores de código mediante inteligencia artificial: un estudio de mapeo sistemático*

*Detecção de odores de código por meio de inteligência artificial: um estudo de mapeamento sistemático*

Enrique Alejandro Chim Mex<sup>1</sup>  
Antonio Armando Aguilera Gomez<sup>2</sup>  
Raul Antonio Aguilar Vera<sup>3</sup>  
Juan Pablo Ucán Pech<sup>4</sup>

**Received:** April 15<sup>th</sup>, 2026

**Approved:** July 31<sup>th</sup>, 2026

**Available:** September 1<sup>st</sup>, 2026

## How to cite this article:

E. A. Chim Mex, A. A. Aguilera Gomez, R. A. Aguilar Vera, and J. P. Ucán Pech, "Detection of Code Smells Using Artificial Intelligence: A Systematic Mapping Study," *Revista Ingeniería Solidaria*, vol. 22, no. 3, 2026.  
doi: <https://doi.org/10.16925/2357-6014.2026.03.12>

---

Research article. <https://doi.org/10.16925/2357-6014.2026.03.12>

- <sup>1</sup> Estudiante de licenciatura en la Facultad de matemáticas, Universidad Autónoma de Yucatán, México.

Email: [a20216391@alumnos.uady.mx](mailto:a20216391@alumnos.uady.mx)

**ORCID:** <https://orcid.org/0009-0005-7579-7370>

- <sup>2</sup> Profesor en la Facultad de matemáticas, Universidad Autónoma de Yucatán, México

Email: [aaguilet@correo.uady.mx](mailto:aaguilet@correo.uady.mx)

**ORCID:** <https://orcid.org/0000-0001-5155-3543>

- <sup>3</sup> Profesor en la Facultad de matemáticas, Universidad Autónoma de Yucatán, México

Email: [avera@correo.uady.mx](mailto:avera@correo.uady.mx)

**ORCID:** <https://orcid.org/0000-0002-1711-7016>

- <sup>4</sup> Profesor en la Facultad de matemáticas, Universidad Autónoma de Yucatán, México.

Email: [juan.ucan@correo.uady.mx](mailto:juan.ucan@correo.uady.mx)

**ORCID:** <https://orcid.org/0000-0002-1013-6396>



## Abstract

*Introduction:* This article is the product of research on code smells developed by the Software Engineering for Education Research Group at the Autonomous University of Yucatán in 2025. This study, like others cited in the literature, has relied on artificial intelligence to expand research in Software Engineering.

*Problem:* Although previous studies have analyzed code smell detection using Machine Learning and Deep Learning independently, this research integrates both approaches to provide a more detailed analysis.

*Objective:* The objective of this study is to present the approach taken toward artificial intelligence and code smells.

*Method:* A systematic mapping study was employed as the research method. Based on a search string executed in five databases—ACM, Taylor & Francis, IEEE, Wiley, and SpringerLink—108 documents were retrieved, of which 35 were selected according to the established inclusion and exclusion criteria.

*Results:* Since 2018, a growing trend in this topic has been observed. Machine Learning is the predominant approach, mainly using the Random Forest algorithm, while CNN stands out in Deep Learning.

*Conclusion:* Despite their growing relevance, there are significant areas that require further attention and research.

*Originality:* This research integrates Machine Learning and Deep Learning approaches within the same study for code smell detection, allowing for a broader view and a more detailed comparison.

*Limitations:* The limitation of this study lies in the fact that the review was based only on five databases and on the inclusion and exclusion criteria.

**Keywords:** Code smell, Machine Learning, Deep Learning, Artificial Intelligence, Software Engineering.

## Resumen

*Introducción:* El artículo es producto de la investigación sobre olores de código desarrollada por el Cuerpo Académico de Ingeniería de Software para la Educación en la Universidad Autónoma de Yucatán en el año 2025. Dicho estudio, como otros citados en la literatura, se ha apoyado en la inteligencia artificial para expandir la investigación en Ingeniería de Software.

*Problema:* Aunque estudios previos han analizado la detección de olores de código utilizando Machine Learning y Deep Learning de manera independiente, esta investigación integra ambos enfoques para ofrecer un análisis más detallado.

*Objetivo:* El objetivo del trabajo es dar a conocer el enfoque que se tiene sobre la inteligencia artificial y los olores de código.

*Método:* Se empleó como método de investigación el estudio de mapeo sistemático. A partir de un string de búsqueda ejecutado en cinco bases de datos —ACM, Taylor & Francis, IEEE, Wiley y SpringerLink— se obtuvieron 108 documentos, de los cuales se seleccionaron 35 con base en los criterios de inclusión y exclusión establecidos.

*Resultados:* Desde 2018, se ha observado una tendencia creciente en el tema. Machine Learning es el enfoque predominante, utilizando principalmente el algoritmo Random Forest, mientras que en Deep Learning se destaca CNN.

*Conclusión:* A pesar de su creciente relevancia, hay áreas significativas que requieren más atención e investigación.

*Originalidad:* Esta investigación integra en un mismo estudio los enfoques de Machine Learning y Deep Learning aplicados a la detección de olores de código, permitiendo una amplia visión y una comparación detallada.

**Limitaciones:** La limitación de este estudio se encuentra en que la revisión se basó únicamente en cinco bases de datos y en los criterios de inclusión y exclusión.

**Palabras clave:** Olor de código, aprendizaje automático, aprendizaje profundo, inteligencia artificial, ingeniería de software.

## Resumo

**Introdução:** Este artigo é resultado de uma pesquisa sobre odores de código desenvolvida pelo Grupo de Pesquisa em Engenharia de Software para a Educação da Universidade Autônoma de Yucatán em 2025. O estudo, assim como outros trabalhos citados na literatura, utiliza inteligência artificial para ampliar a pesquisa em Engenharia de Software.

**Problema:** Embora estudos anteriores tenham analisado a detecção de odores de código utilizando Machine Learning e Deep Learning de forma independente, esta pesquisa integra ambas as abordagens para oferecer uma análise mais detalhada.

**Objetivo:** Apresentar a abordagem adotada em relação à inteligência artificial e aos odores de código.

**Método:** Foi realizado um estudo de mapeamento sistemático. A partir de uma estratégia de busca executada em cinco bases de dados — ACM, Taylor & Francis, IEEE, Wiley e SpringerLink — foram recuperados 108 documentos, dos quais 35 foram selecionados com base nos critérios de inclusão e exclusão estabelecidos.

**Resultados:** Desde 2018, observa-se uma tendência crescente sobre o tema. Machine Learning é a abordagem predominante, utilizando principalmente o algoritmo Random Forest, enquanto, em Deep Learning, destaca-se a CNN.

**Conclusão:** Apesar de sua crescente relevância, existem áreas significativas que requerem maior atenção e pesquisa.

**Originalidade:** Esta pesquisa integra as abordagens de Machine Learning e Deep Learning em um mesmo estudo para a detecção de odores de código, permitindo uma visão mais ampla e uma comparação mais detalhada.

**Limitações:** A revisão foi baseada apenas em cinco bases de dados e nos critérios de inclusão e exclusão.

**Palavras-chave:** Odor de código; aprendizado de máquina; aprendizado profundo; inteligência artificial; engenharia de software.

# 1. INTRODUCTION

In recent years, the number of developers has increased proportionally to market demand in the information technology (IT) industry. Many developers who have recently graduated with a degree related to the IT industry do not have enough experience to produce quality software, which is therefore prone to errors. Practices and tools have been proposed to minimize this problem, such as code reviews and static analysis tools. Despite applying these techniques and tools, something that Fowler defined as a “code smell” may appear at some point during implementation. A code smell (CS) is a characteristic of source code that makes developers sense that the code is not clean. CSs present in the source code can cause risks or failures in the software in the

future [2]. Another point of view is that CSs are design choices made by programmers during the development of the software system that could result in low code maintainability and, therefore, higher maintenance costs [3].

The field of artificial intelligence (AI) goes further: it attempts not only to understand but also to build intelligent systems [4]. There are multiple definitions of AI, including: "The art of creating machines that perform functions that require intelligence when performed by people" [5]. This field is constantly expanding, encompassing various areas such as Machine Learning (ML) and Deep Learning (DL), among others. The purpose of this study is to present the state of the art on the behavior of AI techniques that interact with CSs. Although there are related works, the focus is broadened by understanding the behavior of the two fields (ML and DL), thereby providing new insights into the subject.

## 1.1. RELATED WORKS

Caram et al. [11] conducted a systematic mapping study to understand machine learning patterns in the identification of code smells. This study covers the period from 1999 to 2016, analyzing 26 articles on the subject. They found that the techniques perform similarly to each other, concluding that decision tree, random forest, semi-supervised learning, and nearest-neighbor techniques performed better overall. As for Bayes, they found that it performed worse than the others. Problems were encountered due to the lack of comparability among studies, as each dataset was annotated differently by its authors and did not follow a common standard. Furthermore, they did not report results using the same metrics, which hindered comparisons and the evaluation of the studies, making it difficult to draw conclusions about performance.

Zaidi et al. [12] conducted a systematic mapping study in which they linked artificial intelligence and code smells. The authors analyzed a total of 10 articles selected from five scientific databases. In addition, the study was designed to answer three key research questions. The motivation behind this work was the need to understand how artificial intelligence techniques can be effectively applied to identify code smells. The results obtained demonstrate the relevance of the integration between these two fields, highlighting that ML is the most active model within the field of artificial intelligence in terms of code smells. On the other hand, areas such as computer vision, natural language processing (NLP), robotics, and expert systems showed less attention to this topic by comparison.

Given the authors' motivation on the subject, this article aims to present a systematic mapping study focused on the relationship between artificial intelligence and

CSs. To our knowledge, no study has been found that specifically addresses the five research questions analyzed in 35 articles selected from five scientific databases. Although there is a similar study that addresses artificial intelligence and CSs [12], this study is distinguished by covering a broader period, between 2014 and 2024. This chronological approach allows the remarkable increase in the adoption and evolution of artificial intelligence during that decade to be captured, thereby obtaining a more complete perspective.

The paper is organized as follows: Section 2 describes the research methodology used in the paper. Section 3 presents the results obtained from the mapping. Section 4 presents the discussion, and Section 5 presents the conclusions.

## 2. METHODOLOGY

The Systematic Mapping Study method was chosen for this research. This method allows evidence to be traced in a domain with a high level of granularity [6]. In addition, it helps characterize the information from the studies objectively and systematically [7]. All relevant information is extracted from relevant documents to describe their important aspects. The process carried out is shown in Figure 1 [8].



**Figure 1. Systematic Mapping Process**  
Source: own work

**Planning:** The objective is defined, the research questions to be answered are posed, the databases in which the search will be conducted are specified, and the exclusion and inclusion criteria are established.

**Execution:** The search string is executed in the databases chosen during the planning phase. The documents are evaluated according to the exclusion and inclusion criteria, and the relevant information from the documents is recorded, as well as a history of the search carried out.

**Results:** The results found in the systematic mapping study are presented for discussion and conclusions.

## 2.1. DEFINITIONS OF RESEARCH QUESTIONS

The objective of this study is to provide an overview of the research conducted over the last 10 years on AI and its interaction with CSs. In the context of the above-mentioned objective, the research questions are as follows:

**RQ1:** What is the distribution of articles published on CSs in the last 10 years?

**RQ2:** What types of models are commonly used for CS identification in software development?

This question aims to identify which AI models are used for CS analysis. These include:

**Machine Learning:** Machine Learning is a term directly related to the concept of AI and constitutes a subfield of it, since it refers to the ability of a machine or software system to learn automatically by generating algorithms that respond to certain input data.

**Deep Learning:** The great task of the academic community is to determine algorithms that approximate efficient processing, such as those that simulate brain functions. Advances in DL have generated high computational costs, as they involve the use of multiple parameters. Therefore, one of the main research focuses has been on reducing these costs while maintaining the obtained results.

**Natural Language Processing:** Natural language processing is a subdiscipline of AI and a branch of computational linguistics; its main objective is to build systems and mechanisms that enable communication between people and machines through natural language [13].

**RQ3:** What model techniques are used in CS identification, and which one performs best?

This question aims to identify the techniques used by the models mentioned in RQ2, which are as follows:

### **Machine Learning [14]:**

Classification techniques focus on predicting a qualitative response through data analysis and pattern recognition. There are many different classification techniques or classifiers, the most commonly used being:

- Logistic Regression (LR).
- K-Nearest Neighbors (KNN).
- Support Vector Machine (SVM).

- Random Forest (RF).
- Naive Bayes (NB).
- Multilayer Perceptron (MLP).

### Deep Learning [15]:

This type of machine learning is characterized not only by its depth but also by its ability to learn data representations. The most commonly used techniques within the DL approach are:

- Multilayer Perceptron.
- Convolutional Neural Networks (CNNs).
- Recurrent Neural Networks (RNNs).

**RQ4:** What is the most frequent CS in relation to the level of abstraction in software systems?

The purpose of this question is to identify the most used CS in relation to the level of abstraction. The latter is characterized by the following [16]:

Class level: These usually belong to the design and structure of complete classes, which refers to refactoring or redesigning at the class level.

Method level: These are usually related to the internal implementation and behavior of methods or functions, which refers to refactoring or decomposition of methods.

Table 1 lists the CS at the method level and class level.

**Table 1.** CS by abstraction levels

| Class level              | Method level               |
|--------------------------|----------------------------|
| God Class                | Feature Envy               |
| Data Class               | Long Method                |
| Large Class              | Complex Method             |
| Brain Class              | Empty Catch Block          |
| Lazy Class               | Brain Method               |
| Complex Class            | Complex Conditional        |
| Multifaceted Abstraction | Member Ignoring Method     |
| Misplaced Class          | Internal Getter and Setter |
| Leaking Inner Class      | Long Parameter Lists       |
| Swiss Army Knife         | Magic Number               |

**Source:** own work

**RQ5:** Which datasets are used in the contributions, and what type of data predominates?

## 2.2. IDENTIFICATION AND SELECTION OF RESOURCES

The selection of appropriate resources for searching for relevant literature plays an important role in the analysis [9]. The following resources were selected for the study in order to obtain all the available literature relevant to our research questions.

- ACM (ACM Digital Library).
- Taylor & Francis ( Taylor & Francis Online: Peer-reviewed Journals ).
- IEEE ( IEEE Xplore ).
- Wiley ( Wiley Online Library | Scientific research articles, journals, books, and reference works ).
- SpringerLink ( Home | SpringerLink ).

These databases were chosen because they contain documents related to the field of computing and address information that is relevant to the objective.

## 2.3. CREATION OF SEARCH STRING

The search string was defined using the terms from the research questions combined with the logical operators "AND" and "OR." The terms identified are:

- Code smell: ("code smell" OR "bad smell")
- AI: ("machine learning" OR "deep learning" OR "neural networks" OR "natural language processing" OR "artificial intelligence")
- Level of abstraction: ("application level" OR "class level" OR "method level")

Using logical operators, the resulting search string is shown in Table 2.

**Table 2.** Search string

| Search string                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ((("code smell" OR "bad smell") AND ("machine learning" OR "deep learning" OR "neural networks" OR "natural language processing" OR "artificial intelligence") AND ("application level" OR "class level" OR "method level")) |

**Source:** own work

## 2.4. INCLUSION AND EXCLUSION CRITERIA

Inclusion and exclusion criteria were defined to be applied to the documents resulting from the execution of the string in the databases. These criteria are shown in Table 3.

**Table 3.** Inclusion and Exclusion Criteria

| Criteria | Description                                             |
|----------|---------------------------------------------------------|
| CI1      | All research articles will be included.                 |
| IC2      | Those published between 2014 and 2024 will be included. |
| CI3      | Those in the English language will be included.         |
| CI4      | Those that address the topic will be included.          |
| CE1      | Those that are incomplete will be excluded.             |
| CE2      | Those that are duplicates will be excluded.             |

**Source:** own work

## 2.5. EXECUTION OF THE STRING IN THE DATABASES AND SELECTION OF RESOURCES.

After defining the criteria, the search string was executed across five databases. Figure 2 [9] shows the flow of this activity.

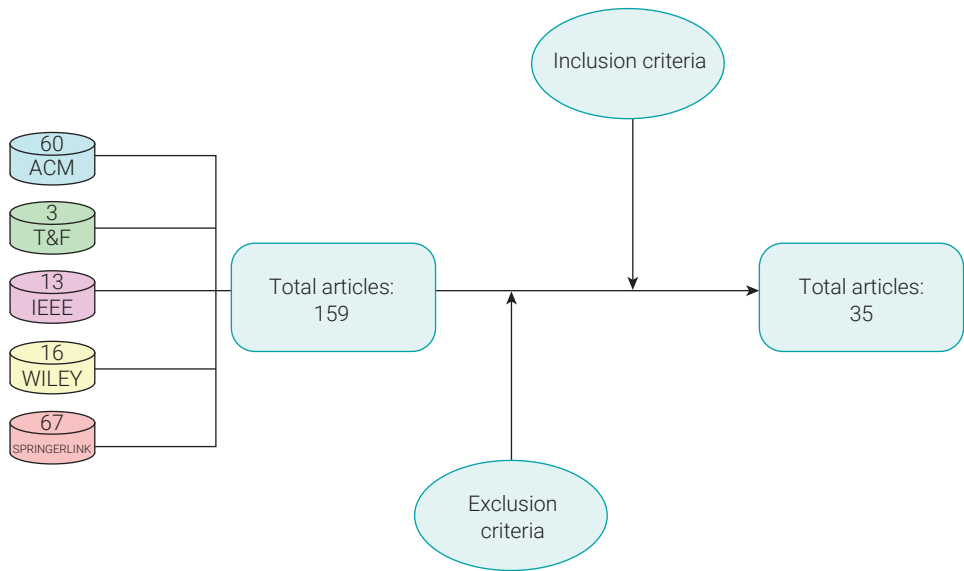


**Figure 2.** Activity flow

**Source:** own work

To select the documents, their titles, abstracts, and keywords were read, and it was also verified that there were no duplicate documents in the total. The history of the search string executions was tracked in an external document. The search performed in the databases yielded a total of 159 documents; after applying the criteria, a total of 35 documents were obtained. Table 4 shows the execution dates of the search string.

**Table 4.** Date of execution of the search string

| Database     | Date       | Total documents |
|--------------|------------|-----------------|
| ACM          | 12/30/2024 | 60              |
| T&F          | 12/30/2024 | 3               |
| IEEE         | 12/30/2024 | 13              |
| WILEY        | 12/30/2024 | 16              |
| SPRINGERLINK | 12/30/2024 | 67              |

**Source:** own work

### 3. RESULTS

The results obtained are presented in the following section, maintaining the focus of the objective set out above.

**RQ1** What is the distribution of articles published on code smells over the last 10 years?

This first question aims to identify the number of contributions from 2014 to 2024. Figure 3 shows the distribution of the corresponding articles in the years mentioned.

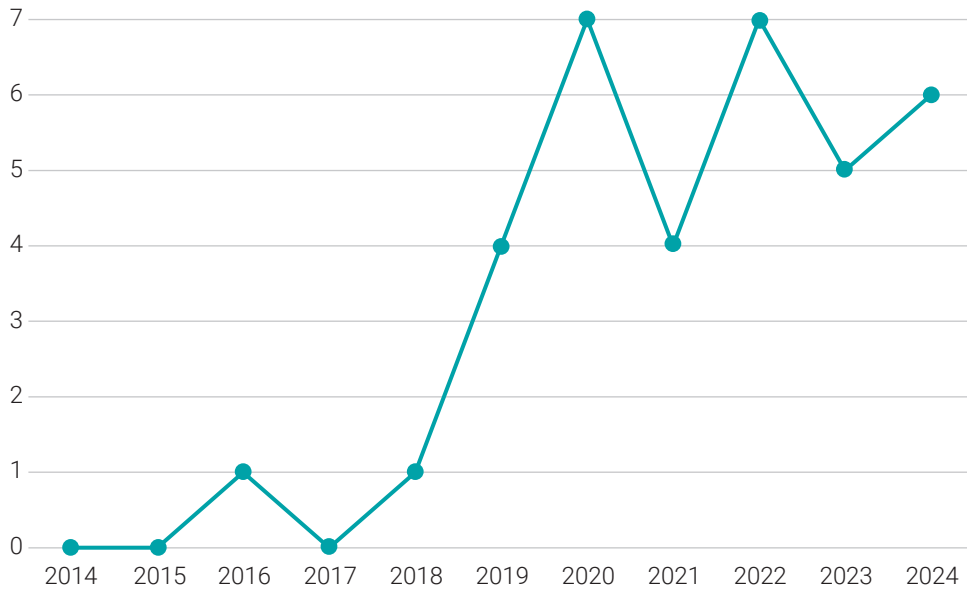


**Figure 3.** Distribution of articles from 2014 to 2024

Source: own work

According to the analysis and contributions of AI interacting with CS, there was an increase in interest in this topic among the community. The analysis begins in 2014. No contributions were made to the topic. In 2015, it can also be seen that no contributions were made. In 2016, the first contribution on the topic was made [20]. In 2017, there were again no contributions.

In 2018, the trend rose again with a single contribution [21]. From this, the topic gained relevance in the following years, with four contributions in 2019 [22][23][24][25]. In 2020, the maximum number of contributions is reached with a total of seven [26][27][28][29][30][31][32]. By 2021, the trend drops again to four contributions [33][34][35][36]. In 2022, there are once again many contributions, with seven [2][38][39][40][41][42][43]. In the following years, the number of contributions decreases, with 5 in 2023 [44][45][46][47][48] and 6 contributions through September 2024 [49][50][18][51][53][17].

**RQ2** What types of models are commonly used to identify code smells in software development?

Figure 4 shows the distribution of models used for CS identification. This distribution allows us to see which type of AI model has been most widely used in the field.

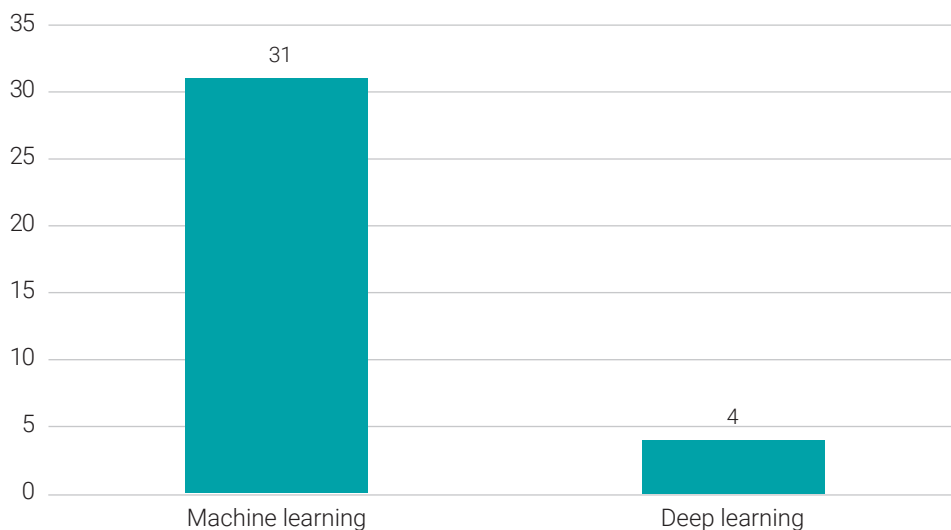


**Figure 4. Distribution of models**

Source: own work

The most representative and widely used model for identifying CS is the ML approach, with a total of 31 contributions, indicating a preference for the use of this technique. Likewise, the least representative and least used model is DL, which has a total of 4 contributions to the scientific community.

**RQ3** Which model techniques are used in the identification of CS, and which one showed the best performance?

In order to answer the question, the techniques used in the contributions were considered. The contributions used between 1 and 6 techniques for CS detection. The distributions of the techniques used in the ML approach are shown in Figure 5. The distributions of the techniques used in the DL approach are shown in Figure 6.

Regarding the ML approach, it can be seen that the RF technique is the most used with a total of 26 occurrences. Similar findings have been reported in studies published in this journal, where RF is frequently employed for predictive tasks [52]. Followed by Naive Bayes used a total of 17 times, followed by DT used 15 times, KNN and SVM used 13 times, RL and MLP used 11 times, J48 with a total of 6, Gradient Boosting Machine (GBM) with a total of 3, Gradient Boosted Tree (GBT) with a total of 2, and other techniques used once.

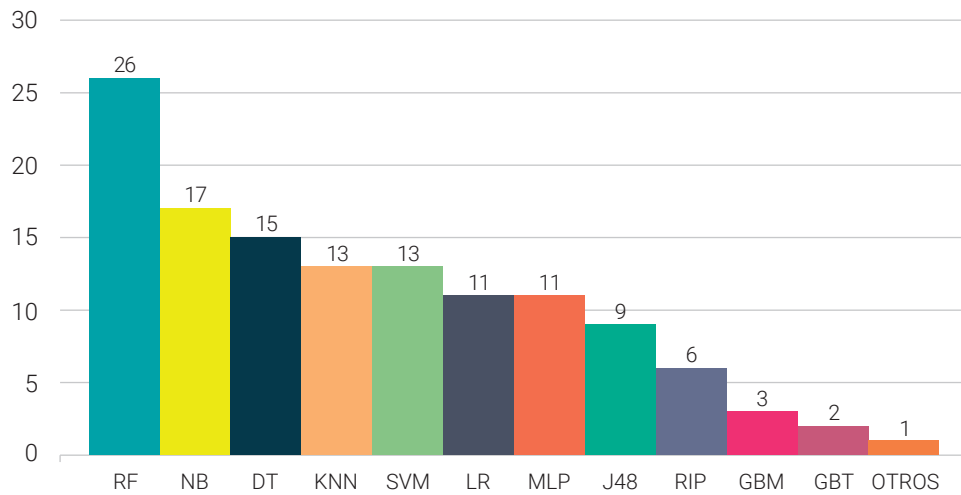


**Figure 5. ML techniques**

Source: own work

Regarding the ML approach, the RF technique is the most used, appearing a total of 26 times in the contributions. It is followed by NB, which has a total of 17 times in the contributions, followed by DT, which is used 15 times. KNN and SVM occupy the same place, each appearing a total of 13 times in the contributions.

Other techniques such as MLP and RL were used 11 times respectively. On the other hand, J48 and Jrip were used 9 and 6 times respectively, GBM was used 3 times, and GBT on 2 occasions. Other ML techniques that were used only once were also analyzed.

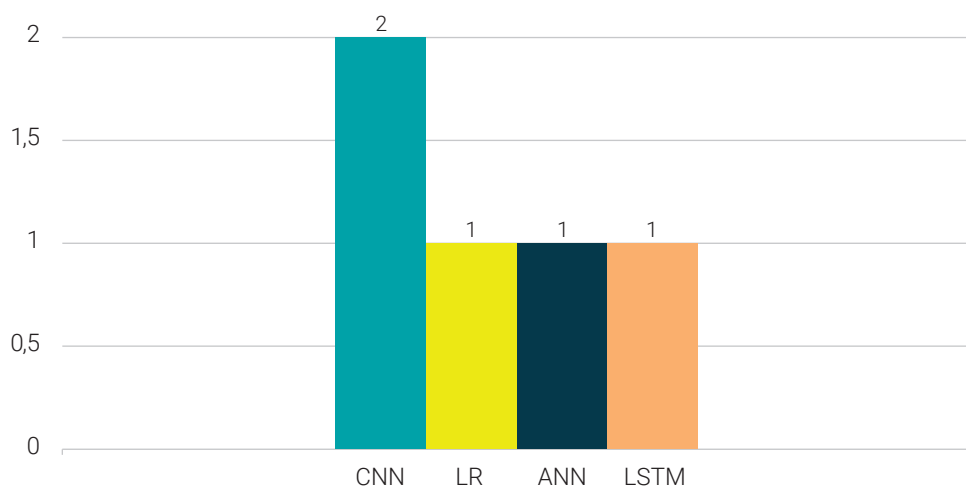


**Figure 6. DL techniques**

Source: own work

For the DL approach, the most widely used technique is Convolutional Neural Network (CNN), appearing twice in the contributions. It is followed by RL, Artificial Neural Network (ANN), and Long Short-Term Memory (LSTM), each appearing once in the contributions. Each of these techniques provides different approaches to CS detection.

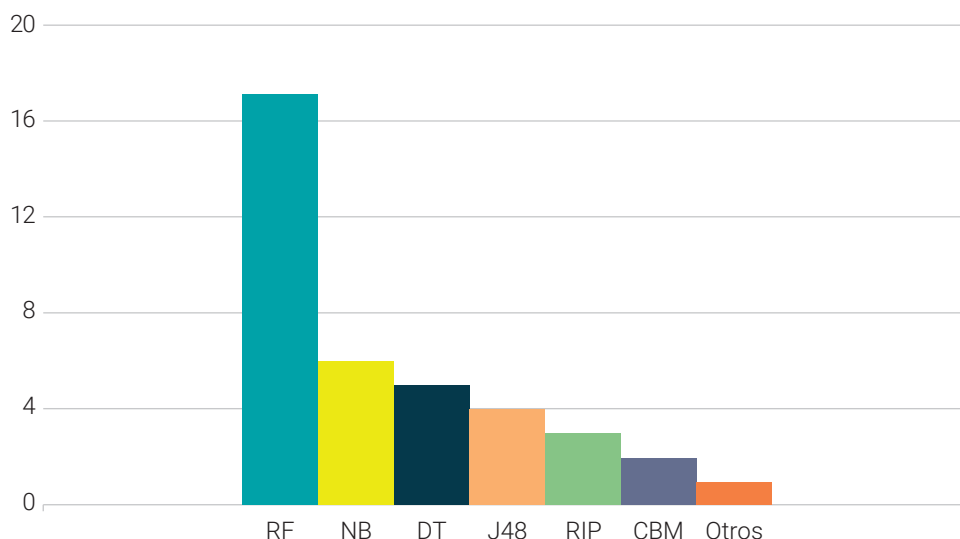


**Figure 7.** Algorithm with the highest performance

Source: own work

Figure 7 shows that the algorithm with the best performance in CS detection was RF, which obtained the best results on 17 occasions. In second place is NB, demonstrating considerable performance on six occasions, followed by DT, which stood out on five occasions.

Subsequently, J48 demonstrated good results on 4 occasions, while JRIP did so on 3 occasions. Finally, GBM demonstrated great performance on 2 occasions. Likewise, other algorithms were identified that were most effective on a single occasion.

It is important to note that the algorithms that demonstrated the best performance at their level of abstraction were selected.

**RQ4** What is the most frequent code smell in relation to the level of abstraction in software systems?

The fourth question aims to identify the most used CS in contributions at the abstraction level. Figure 8 shows the most used CS at the class level. Figure 9 shows the most used CS at the method level.

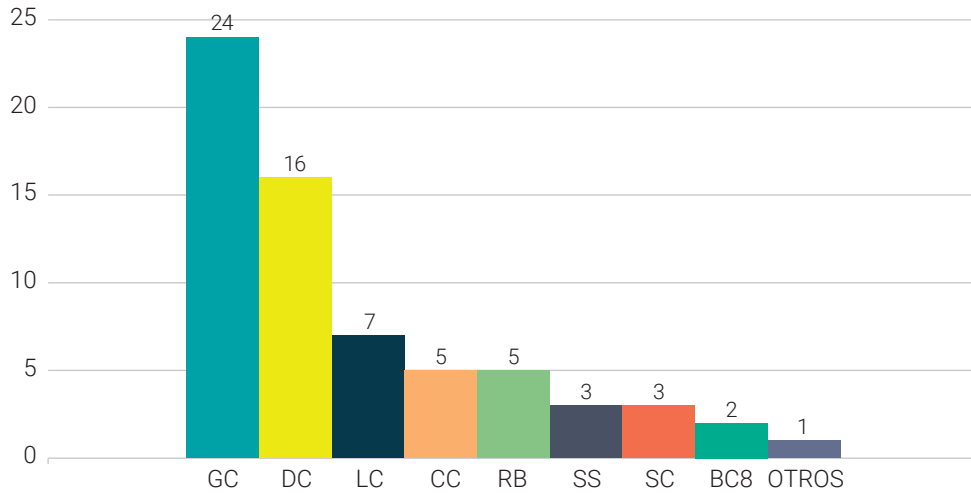


**Figure 8. CS at the class level**

Source: own work

In the analysis of the level of abstraction at the class level, a trend toward the CS known as God Class was identified, which appeared 24 times in the contributions. This was followed by the CS data class, which appeared 16 times in the contributions. On the other hand, Large Class was recorded 7 times.

As for complex class and Refused Bequest, both tied with five appearances each. Shotgun Surgery and Spaghetti Code appeared three times. Blob Class was mentioned only twice. Finally, other CS were identified, which, although less frequent, appeared only once in the contributions.

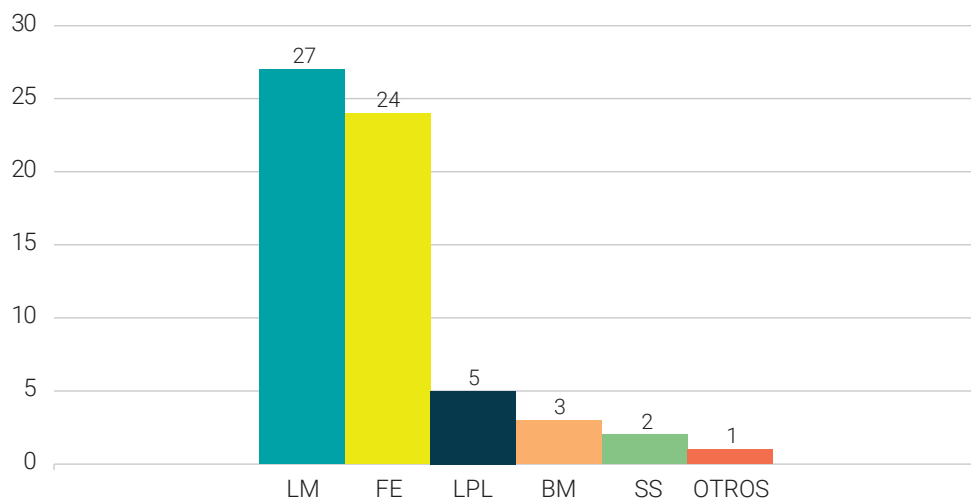


**Figure 9. CS at the method level**

Source: own work

In the analysis of the level of abstraction at the method level, a marked trend toward the Long Method CS was identified, which appeared 27 times in the contributions. It was followed by Feature Envy, which appeared 24 times in the contributions.

On the other hand, Long Parameter List appeared 5 times in the contributions, and the Brain Method appeared 3 times, while Switch Statements appeared twice. In addition, other CS were identified, although less frequently, each appearing only once in the contributions.

**RQ5** What datasets are used in the contributions and what type of data predominates?

Figure 10 shows the trend observed in the use of dataset for SC analysis. A total of 19 studies chose to develop their own dataset, obtaining the related information from previously created project repositories on GitHub.

The Qualitas Corpus dataset ranks second, having been used by 13 studies. This dataset is a collection of software systems intended for use in empirical studies. It contains 112 systems, 15 systems with 10 or more versions, and 754 versions in total.

It should be noted that other datasets with little use were identified. The LandFill dataset was used by 2 studies, and the Multi-Label dataset was used by 1 study. These datasets show less use compared to the datasets created by the authors and the Qualitas Corpus dataset.

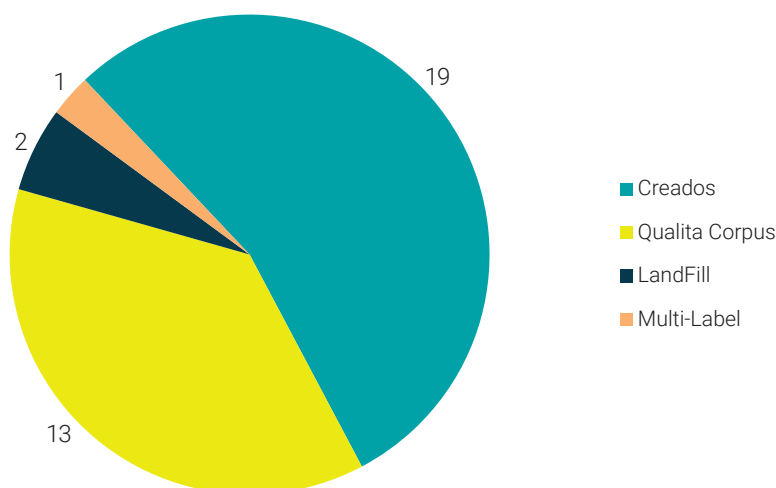


**Figure 10. Distribution of dataset**  
Source: own work

## 4. DISCUSSION

In this systematic mapping study, contributions from different authors addressing the topic of CS and artificial intelligence, published over a 10-year period from 2014 to 2024, were selected. The results of RQ1 show an incremental trend in contributions since 2018, reaching their first peak in 2020. However, in 2021, an opposite trend is observed. Nevertheless, this trend reverses in 2022, when a new incremental peak occurs. Subsequently, in 2023, there is a new decline, followed by a rebound in 2024, when contributions again reach a notable peak. This growth may be driven by the pursuit of higher software quality. Among the motivations, there is a notable interest in using machine learning classifiers, such as RF, SVM, and KNN, as well as specialized tools and advanced techniques for CS detection, including CNNs and GNNs [27], [40].

Regarding the models used for CS detection, the results of RQ2 show that ML was the most favored approach, being used in 31 contributions. Furthermore, it has been key in the construction of databases and the evaluation of classifiers to meet different requirements [2]. Similarly, DL was used in four contributions, reflecting a lower degree of use compared to ML. Likewise, no contributions related to NLP were found, which could be due to the fact that the databases used are composed of numerical data. This indicates that there is a large area of opportunity within this field. As shown by the results of RQ3, RF is the most widely used classification technique among traditional algorithms in contributions related to ML. This could be because this technique has shown good results, as observed in the results of RQ4. On the other hand, in terms of DL techniques and given the number of contributions, a preference for CNNs could be observed.

The contributions analyzed different CS types according to their level of abstraction, as shown in the results of RQ5. For the class-level abstraction type, there is a preference for the God Class code smell, followed by Data Class, these being the most representative in the contributions. Other class-level code smells were observed that were used only once in the contributions. In terms of method-level code smells, there was a preference for the Long Method code smell, followed by Feature Envy, these being the most representative. It should be noted that other method-level code smells were observed that were used only once.

The results obtained indicate a growing interest among the scientific community in detecting CSs using AI. This trend highlights the potential of source code automation and its impact on improving software quality while reducing process complexity.

## 5. CONCLUSION

This study is characterized by the detection of CSs using AI through a systematic mapping study of scientific and relevant articles on the subject published from 2014 to 2024. Through this study, it was possible to identify various ML and DL techniques used in the identification and classification of CSs. It also allowed the identification of the different types of CSs that occur in software projects, classifying them according to their level of abstraction (class and method). In addition, the datasets used by the authors in their contributions were identified. Finally, the different techniques for handling data imbalance, a major challenge in CS detection, were analyzed.

## REFERENCES

- [1] M. Jerzyk and L. Madeyski, "Code smells: A comprehensive online catalog and taxonomy," in *Studies in Systems, Decision and Control*. Cham, Switzerland: Springer Nature Switzerland, 2023, pp. 543–576, doi: [https://doi.org/10.1007/978-3-031-25695-0\\_24](https://doi.org/10.1007/978-3-031-25695-0_24).
- [2] B. T. Nguyen, N. H. M. Nguyen, T. M. H. Le, and B. T. Nguyen, "ml-Codesmell: A code smell prediction dataset for machine learning approaches," in *Proc. 11th Int. Symp. Information and Communication Technology (SolCT)*, Hanoi, Vietnam. New York, NY, USA: ACM, 2022, pp. 368–374, doi: <https://doi.org/10.1145/3568562.3568643>.
- [3] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, and A. De Lucia, "A large-scale empirical study on the lifecycle of code smell co-occurrences," *Information and Software Technology*, vol. 99, pp. 1–10, Jul. 2018, doi: <https://doi.org/10.1016/j.infsof.2018.02.004>.
- [4] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Harlow, U.K.: Pearson Education, 2010.
- [5] J. Haugeland, *Artificial Intelligence: The Very Idea*. Cambridge, MA, USA: MIT Press, 1989, doi: <https://doi.org/10.7551/mitpress/1170.001.0001>.
- [6] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University, Keele, U.K., Tech. Rep. EBSE-2007-01, 2007. [Online]. Available: (SD)
- [7] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, "Using mapping studies in software engineering," in *Proc. 20th Psychology of Programming Interest Group (PPIG) Conf.*, Lancaster, U.K., 2008, pp. 195–204.

- [8] A. A. Aguilera and O. S. Gómez, "Software engineering research in Mexico: A systematic mapping study," *International Journal of Software Engineering and Its Applications*, vol. 10, no. 12, pp. 75–92, Dec. 2016, doi: <https://doi.org/10.14257/ijseia.2016.10.12.07>.
- [9] M. I. Azeem, F. Palomba, L. Shi, and Q. Wang, "Machine learning techniques for code smell detection: A systematic literature review and meta-analysis," *Information and Software Technology*, vol. 108, pp. 115–138, Apr. 2019, doi: <https://doi.org/10.1016/j.infsof.2018.12.009>.
- [10] F. Arcelli Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1143–1191, 2016, doi: <https://doi.org/10.1007/s10664-015-9378-4>.
- [11] F. L. Caram, B. R. D. O. Rodrigues, A. S. Campanelli, and F. S. Parreiras, "Machine learning techniques for code smells detection: A systematic mapping study," *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 2, pp. 285–316, 2019, doi: <https://doi.org/10.1142/S021819401950013X>.
- [12] M. A. Zaidi and R. Colomo-Palacios, "Code smells enabled by artificial intelligence: A systematic mapping," in *Computational Science and Its Applications–ICCSA 2019*. Cham, Switzerland: Springer, 2019, pp. 418–427, doi: [https://doi.org/10.1007/978-3-030-24305-0\\_31](https://doi.org/10.1007/978-3-030-24305-0_31).
- [13] D. F. Escobar Cerón and A. D. Quevedo Cortés, "Inteligencia artificial como alternativa en la detección de noticias falsas," *Tecnología, Investigación y Academia (TIA)*, vol. 10, no. 1, pp. 20–37, Aug. 2022. (SD)
- [14] I. N. Salamanca Rativa, "Técnicas de aprendizaje automático aplicadas en los sistemas de predicción," *Tecnología, Investigación y Academia (TIA)*, vol. 8, no. 1, pp. 37–53, Jul. 2021. (SD)
- [15] A. González-Muñiz, "Applications of artificial intelligence techniques based on deep learning to the analysis and improvement of industrial process efficiency," Ph.D. dissertation, Universidad de Oviedo, Oviedo, Spain, 2018, doi: <https://doi.org/10.13140/RG.2.2.15144.72967>.
- [16] F. Zhang *et al.*, "Data preparation for deep learning-based code smell detection: A systematic literature review," *Journal of Systems and Software*, vol. 216, Art. no. 112131, 2024, doi: <https://doi.org/10.1016/j.jss.2024.112131>.
- [17] M. Hadj-Kacem and N. Bouassida, "Multi-label learning for identifying co-occurring class code smells," *Computing*, vol. 106, pp. 2585–2612, 2024, doi: <https://doi.org/10.1007/s00607-024-01294-x>.

- [18] A. Alazba, H. Aljamaan, and M. Alshayeb, “CoRT: Transformer-based code representations with self-supervision by predicting reserved words for code smell detection,” *Empirical Software Engineering*, vol. 29, no. 3, Art. no. 57, 2024, doi: <https://doi.org/10.1007/s10664-024-10445-9>.
- [19] H. A. Arnejo Calviño, “Métodos para la mejora de predicciones en clases desbalanceadas en el estudio de bajas de clientes (churn),” M.S. thesis, Máster en Técnicas Estadísticas, Universidad de Santiago de Compostela, Santiago de Compostela, Spain, 2017.
- [20] F. Arcelli Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, “Comparing and experimenting machine learning techniques for code smell detection,” *Empirical Software Engineering*, vol. 21, no. 3, pp. 1143–1191, 2016, doi: <https://doi.org/10.1007/s10664-015-9378-4>.
- [21] K. Karaduzovic-Hadziabdic and R. Spahic, “Comparison of machine learning methods for code smell detection using reduced features,” in *Proc. 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, Bosnia and Herzegovina, Sep. 20–23, 2018, pp. 670–672, doi: <https://doi.org/10.1109/UBMK.2018.8566561>.
- [22] F. Pecorelli, D. Di Nucci, C. De Roover, and A. De Lucia, “On the role of data balancing for machine learning-based code smell detection,” in *Proc. 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, Tallinn, Estonia, Aug. 27, 2019, pp. 19–24, doi: <https://doi.org/10.1145/3340482.3342744>.
- [23] F. C. Luiz, B. R. de Oliveira Rodrigues, and F. S. Parreiras, “Machine learning techniques for code smells detection,” in *Proc. XV Brazilian Symposium on Information Systems (SBSI)*, Aracaju, Brazil, 2019, pp. 1–8, doi: <https://doi.org/10.1145/3330204.3330275>.
- [24] E. O. Kiyak, D. Birant, and K. U. Birant, “Comparison of multi-label classification algorithms for code smell detection,” in *Proc. 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Ankara, Türkiye, Oct. 11–13, 2019, pp. 1–6, doi: <https://doi.org/10.1109/ISMSIT.2019.8932855>.
- [25] M. Hadj-Kacem and N. Bouassida, “Deep representation learning for code smells detection using variational auto-encoder,” in *Proc. International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, Jul. 14–19, 2019, pp. 1–8, doi: <https://doi.org/10.1109/IJCNN.2019.8851854>.
- [26] F. Pecorelli, F. Palomba, F. Khomh, and A. De Lucia, “Developer-driven code smell prioritization,” in *Proc. 17th International Conference on Mining Software Repositories (MSR)*, Seoul, Republic of Korea, 2020, pp. 220–231, doi: <https://doi.org/10.1145/3379597.3387457>.

- [27] S. Lujan, F. Pecorelli, F. Palomba, A. De Lucia, and V. Lenarduzzi, "A preliminary study on the adequacy of static analysis warnings with respect to code smell prediction," in *Proc. 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Virtual Event, USA, 2020, pp. 1–6, doi: <https://doi.org/10.1145/3416505.3423559>.
- [28] D. Cruz, A. Santana, and E. Figueiredo, "Detecting bad smells with machine learning algorithms," in *Proc. International Conference on Technical Debt (TechDebt)*, Seoul, Republic of Korea, 2020, pp. 31–40, doi: <https://doi.org/10.1145/3387906.3388618>.
- [29] R. Lima et al., "Understanding and detecting harmful code," in *Proc. 34th Brazilian Symposium on Software Engineering (SBES)*, Natal, Brazil, 2020, pp. 223–232, doi: <https://doi.org/10.1145/3422392.3422420>.
- [30] S. Jain and A. Saha, "Rank-based univariate feature selection methods on machine learning classifiers for code smell detection," *Evolutionary Intelligence*, vol. 15, no. 1, pp. 609–638, 2022, doi: <https://doi.org/10.1007/s12065-020-00536-z>.
- [31] M. Y. Mhawish and M. Gupta, "Predicting code smells and analysis of predictions: Using machine learning techniques and software metrics," *Journal of Computer Science and Technology*, vol. 35, no. 6, pp. 1428–1445, Nov. 2020, doi: <https://doi.org/10.1007/s11390-020-0323-7>.
- [32] T. Guggulothu and S. A. Moiz, "Code smell detection using multi-label classification approach," *Software Quality Journal*, vol. 28, no. 3, pp. 1063–1086, 2020, doi: <https://doi.org/10.1007/s11219-020-09498-y>.
- [33] A. Patnaik and N. Padhy, "Does code complexity affect the quality of real-time projects?," in *Proc. International Conference on Data Science, Machine Learning and Artificial Intelligence (DSMLAI)*, Windhoek, Namibia, 2021, pp. 178–185, doi: <https://doi.org/10.1145/3484824.3484911>.
- [34] H. Aljamaan, "Voting heterogeneous ensemble for code smell detection," in *Proc. 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Pasadena, CA, USA, Dec. 13–16, 2021, pp. 897–902, doi: <https://doi.org/10.1109/ICMLA52953.2021.00148>.
- [35] Y. Zhang and C. Dong, "MARS: Detecting brain class/method code smell based on metric–attention mechanism and residual network," *Journal of Software: Evolution and Process*, vol. 34, no. 5, Art. no. e2403, 2022, doi: <https://doi.org/10.1002/smr.2403>.
- [36] A. Gupta and N. K. Chauhan, "A severity-based classification assessment of code smells in Kotlin and Java applications," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1831–1848, 2022, doi: <https://doi.org/10.1007/s13369-021-06077-6>.

- [37] N. Vatanapakorn, C. Soomlek, and P. Seresangtakul, "Python code smell detection using machine learning," in *Proc. 26th International Computer Science and Engineering Conference (ICSEC)*, Sakon Nakhon, Thailand, Dec. 21–23, 2022, pp. 128–133, doi: <https://doi.org/10.1109/ICSEC56337.2022.10049330>.
- [38] A. Abdou and N. Darwish, "Severity classification of software code smells using machine learning techniques: A comparative study," *Journal of Software: Evolution and Process*, vol. 35, no. 2, Art. no. e2454, 2023, doi: <https://doi.org/10.1002/smr.2454>.
- [39] J. Nanda and J. K. Chhabra, "SSHM: SMOTE-stacked hybrid model for improving severity classification of code smell," *International Journal of Information Technology*, vol. 14, no. 5, pp. 2701–2707, 2022, doi: <https://doi.org/10.1007/s41870-022-00943-8>.
- [40] F. Pecorelli, S. Lujan, V. Lenarduzzi, F. Palomba, and A. De Lucia, "On the adequacy of static analysis warnings with respect to code smell prediction," *Empirical Software Engineering*, vol. 27, no. 3, Art. no. 61, 2022, doi: <https://doi.org/10.1007/s10664-022-10126-5>.
- [41] S. Singh, M. Jalal, and S. Kaur, "Bug classification depend upon refactoring area of code," *Journal of The Institution of Engineers (India): Series B*, vol. 104, no. 1, pp. 61–77, 2023, doi: <https://doi.org/10.1007/s40031-022-00836-1>.
- [42] J. P. dos Reis, F. Brito e Abreu, and G. de F. Carneiro, "Crowdsampling: A preliminary study on using collective knowledge in code smells detection," *Empirical Software Engineering*, vol. 27, no. 3, Art. no. 58, 2022, doi: <https://doi.org/10.1007/s10664-021-10110-5>.
- [43] P. Sukkasem and C. Soomlek, "Enhanced machine learning-based code smell detection through hyper-parameter optimization," in *Proc. 20th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Phitsanulok, Thailand, Jun. 28–Jul. 1, 2023, pp. 297–302, doi: <https://doi.org/10.1109/JCSSE58229.2023.10202124>.
- [44] F. Li, K. Zou, J. W. Keung, X. Yu, S. Feng, and Y. Xiao, "On the relative value of imbalanced learning for code smell detection," *Software: Practice and Experience*, vol. 53, no. 10, pp. 1902–1927, 2023, doi: <https://doi.org/10.1002/spe.3235>.
- [45] R. S. Rao, S. Dewangan, A. Mishra, and M. Gupta, "A study of dealing class imbalance problem with machine learning methods for code smell severity detection using PCA-based feature selection technique," *Scientific Reports*, vol. 13, no. 1, Art. no. 16277, 2023, doi: <https://doi.org/10.1038/s41598-023-43380-8>.

- [46] S. Dewangan, R. S. Rao, S. R. Chowdhuri, and M. Gupta, "Severity classification of code smells using machine-learning methods," *SN Computer Science*, vol. 4, no. 5, Art. no. 537, 2023, doi: <https://doi.org/10.1007/s42979-023-01979-8>.
- [47] A. Kovačević *et al.*, "Automatic detection of code smells using metrics and CodeT5 embeddings: A case study in C#," *Neural Computing and Applications*, vol. 36, no. 16, pp. 9203–9220, 2024, doi: <https://doi.org/10.1007/s00521-024-09551-y>.
- [48] H. G. Nunes, A. Santana, E. Figueiredo, and H. Costa, "Tuning code smell prediction models: A replication study," in *Proc. IEEE/ACM 32nd International Conference on Program Comprehension (CPC)*, Lisbon, Portugal, 2024, pp. 316–327, doi: <https://doi.org/10.1145/3643916.3644436>.
- [49] P. S. Yadav, R. S. Rao, and A. Mishra, "An evaluation of multi-label classification approaches for method-level code smells detection," *IEEE Access*, vol. 12, pp. 53664–53676, 2024, doi: <https://doi.org/10.1109/ACCESS.2024.3387856>.
- [50] A. Nandini, R. Singh, and A. Rathee, "Improving code smell detection by reducing dimensionality using ensemble feature selection and machine learning," *SN Computer Science*, vol. 5, no. 6, Art. no. 702, 2024, doi: <https://doi.org/10.1007/s42979-024-03013-x>.
- [51] D. Wu *et al.*, "iSMELL: Assembling LLMs with expert toolsets for code smell detection and refactoring," in *Proc. IEEE/ACM 39th International Conference on Automated Software Engineering (ASE)*, Sacramento, CA, USA, 2024, pp. 1345–1357, doi: <https://doi.org/10.1145/3691620.3695508>.
- [52] L. Arenales, V. Villarreal, and J. J. Saldaña-Barrios, "Predictive model based on artificial intelligence and contextual awareness to identify students at risk of dropping out of university in Panama," *Revista Ingeniería Solidaria*, vol. 21, no. 1, pp. 1–23, 2025, doi: <https://doi.org/10.16925/2357-6014.2025.01.08>.