Can Public Code Smells Datasets Be Trusted?

Ruchin Gupta, Jitendra Kumar Seth, Anupama Sharma, and Abhishek Goyal

Original scientific article

***Abstract*—Code smells signal potential issues in a codebase and indicate technical debt. Early detection is crucial for maintaining code quality. Researchers often rely on public datasets to automate and enhance smell detection, but their trustworthiness is frequently assumed rather than verified. While these datasets are valuable for developing detection tools, key questions arise: Can they be fully trusted? Are the labels accurate? Do they reflect real-world software development? Recent studies reveal inconsistencies, biases, and misclassifications, raising concerns about their reliability. This paper explores the integrity of widely used 2 sets of public code smells datasets namely Group A dataset and Group B dataset by examining their internal consistency, alignment with established facts. Through this investigation, we aim to determine whether these datasets can be confidently utilized in research and practical applications, or if their inherent issues undermine the validity of the results they produce. Group A datasets are smaller, balanced, and factually aligned but lack industry relevance, while Group B deviates from known facts. The study acknowledges academic–industry differences, viewing divergence as a reflection of real-world variability rather than a flaw, and emphasizes the need for rigorous validation of public datasets to ensure reliable research outcomes.**

Keywords*—*Code smell, code smell datasets, validation.

I. Introduction

The concept of code smells was first introduced by Fowler and Beck in 1999 [1], referring to symptoms in the source code that may indicate deeper issues within the software design. Since then, a wide array of tools and techniques have been developed to automatically detect these smells, enabling software engineers to identify and address maintainability problems early on. To facilitate the development and evaluation of smell detection tools, public code smell datasets have been constructed as critical resources for the research community.

One of the most widely known datasets is from the PROMISE repository, which contains various labelled datasets used for code smell detection, software defect prediction, and other software quality assessment tasks. These datasets have been used in numerous studies, providing a common basis for evaluating detection methods. Similarly, the Qualitas Corpus and SDP Repository also, house datasets utilized for smell detection across different software systems.

Manuscript received September 23, 2025; revised October 20, 2025. Date of publication December 15, 2025. Date of current version December 15, 2025. The associate editor prof. Tihana Galinac Grbac has been coordinating the review of this manuscript and approved it for publication.

R. Gupta is with the Jaypee Institute of Information Technology, Noida, India (e-mail: skg11in@yahoo.co.in).

J. K. Seth and A. Goyal are with the KIET Group of Institutions, Delhi-NCR, Ghaziabad, India (e-mails: drjkseth@gmail.com, abhi.goyal99@gmail.com).

A. Sharma is with the Ajay Kumar Garg Engineering College, Ghaziabad, India (e-mail: anupama0027@gmail.com).

Digital Object Identifier (DOI): 10.24138/jcomss-2025-0131

Many researchers, including Hall et al. [2] and Palomba et al. [3], have utilized these repositories for empirical studies on code smells, leading to the development of sophisticated detection tools like JDeodorant and PMD.

Despite their utility, public datasets on code smells are not without limitations. Researchers have raised concerns about the variability in the definition and labelling of code smells across datasets. For instance, Pietrzak and Walter [4] analysed relations among different types of smells and proposed refined definitions to improve the consistency of detection. However, even with clearer definitions, inconsistencies in smell annotations persist across different datasets, complicating efforts to build generalized models for smell detection.

Another challenge noted in the literature is the dependency of smell annotations on individual reviewers. Yamashita and Moonen [5] conducted a large-scale empirical study that found substantial variations in how human reviewers identify code smells, leading to disagreements in the labelling process. This introduces subjectivity into the datasets, which could influence the performance and evaluation of detection tools built on these datasets. The variability in labelling is especially problematic in cross-project and cross-domain studies, where the transferability of smell detection models can be hampered by dataset-specific biases.

Recent studies have also examined the trustworthiness of public datasets by analysing the extent to which these datasets reflect real-world code smells. Palomba et al. [6] explored the impact of refactoring efforts on the presence of smells in open-source projects and found discrepancies between labelled smells and the actual structure of the code after refactoring. Similarly, research by Fontana et al. [7] and Tufano et al. [8] highlighted the need for more rigorous validation of smell annotations, noting that inconsistencies in these datasets could lead to misleading conclusions in empirical studies.

The paper [9] presents DACOS, a dataset with 10,267 annotations across 5,192 code snippets, focusing on three code smells: multifaceted abstraction, complex method, and long parameter list. The dataset was built in two phases: first, identifying subjective code snippets using metric thresholds, and second, collecting manual annotations for these snippets. The authors also introduce DACOSX, an extended version that includes both clearly smelly and non-smelly code snippets. To support annotation, they developed TagMan, a web tool for systematic labelling. The dataset and tool are publicly available to aid in developing machine-learning models for code smell detection.

The study [10] introduced a dataset featuring two Python code smells: Large Class and Long Method. To create this dataset, the metrics and open-source Python projects, extracted relevant source code metrics, and utilized the PySmell tool to

identify smelly samples. The dataset comprises 200 smelly and 800 non-smelly instances for each code smell.

In their systematic literature review, Zhang et al. [11] analysed 36 studies on deep learning-based code smell detection (CSD) and highlighted several challenges related to code smell datasets: (a) Data Diversity: Existing datasets often lack diversity, limiting the generalizability of CSD models. (b)Standardization: There is an absence of standardized data formats and labeling conventions, hindering effective model training and evaluation, (c) Accessibility: Many datasets are not publicly accessible, restricting their use in research and industry.

According to the study [12], the dataset proposed by Palomba et al. [6], along with the MLQC dataset introduced by Madeyski et al. [13], stands out as one of the most comprehensive publicly available code smell datasets, particularly in terms of size and data quality. However, the dataset by Palomba et al. [6] only contains samples.

Beyond dataset quality, several scholars have emphasized the importance of dataset diversity. Maldonado et al. [14] stressed the need for datasets that cover a broad spectrum of programming languages, paradigms, and software domains to ensure the generalizability of smell detection techniques. However, many available datasets focus predominantly on Java projects, limiting their applicability to other languages or development contexts.

Given these challenges, recent efforts have sought to improve the quality and reliability of code smell datasets. For example, studies have proposed automated approaches to minimize human bias in smell labelling, including the use of machine learning models trained on well-curated datasets. Other initiatives have focused on developing standard metrics for evaluating the consistency and accuracy of smell datasets, as seen in the work of Khomh et al. [15], who advocated for more transparent and reproducible dataset creation processes.

Despite these efforts, there is still no consensus on the best practices for curating and validating public code smell datasets. The ongoing concerns about data quality, bias, and applicability suggest that these datasets require further scrutiny to ensure their trustworthiness in both academic research and industrial applications. To address this research gap, we propose an approach for the first time (to the best of our knowledge) analysing publicly available code smell datasets.

The study makes the following key contributions:

- Validation of Code Smell Datasets: We present a novel method to evaluate the validity and reliability of publicly available code smell datasets, with a particular focus on analysing code smell relationships. Our study does not claim an absolute ground truth but instead evaluates internal consistency and alignment with established patterns as a way to assess relative trustworthiness, not correctness in an absolute sense.
- Comprehensive Analysis of Popular Datasets: For experimentation, we analysed two set of widely recognized datasets [16], [17] in depth. The study conducted by Fontana et al. [16] is a crucial and notable advancement in the domain of code smell. The study also uses industry-relevant code smell datasets [17] and constructs new datasets using different approaches. This is the largest publicly available dataset featuring manually validated labels. The use of Industry-relevant datasets ensures the relevance and practical applicability of the findings. The results were rigorously compared against established findings in the literature, highlighting discrepancies and insights.

The following text presents the structure of the article. Section II describes the methodology for assessing the trustworthiness of datasets. Section III details the selection criteria and datasets used in the study. Section IV analyses the constructed multilabel datasets to examine smell interactions. Finally, Section V summarizes the conclusions and highlights potential directions for future research.

II. Method for Analysing the Trustworthiness of Code Smell Datasets

Ensuring the reliability of publicly available code smell datasets is crucial for advancing research in software quality assessment. This study proposes a method to evaluate the trustworthiness of such datasets by analysing the interactions between different code smells and contrasting these interactions with established theoretical and empirical facts.

The methodology consists of the following key steps:

Step 1. Dataset Selection and Pre-processing - The datasets were chosen based on three criteria: (a) public accessibility, (b) prior use in peer-reviewed research, and (c) availability of manual validation. Together, these ensure both reproducibility and representativeness. Each dataset contains information about the presence of various code smells in different software systems. The datasets were pre-processed to ensure consistency, removing duplicate entries, wherever necessary. The selected datasets correspond to four distinct code smells—Long Method (LM), God Class (GC), Data Class (DC), and Feature Envy (FE) that are widely recognized and frequently studied in the literature.

Step 2. Development of multilabel dataset - To investigate the interaction between different code smells, pairs of code smells were multilabel into a single dataset. This process involved mapping occurrences of two selected code smells within the same codebase or software module/sample. The objective was to identify co-occurrences and patterns that suggest potential relationships between the smells.

Step 3. Analysis of multilabel datasets for smell Interaction Analysis - The multilabel datasets were analysed to detect correlations and dependencies between the code smells. The technique of association rule mining was employed to quantify the strength of these interactions. High or low correlation values were used as indicators of whether two code smells are likely to co-exist in a meaningful way.

Step 4. Contrasting with Established Facts - The observed interactions were compared against well-established software engineering principles and empirical findings from previous studies. This step involved validating whether the detected relationships align with known theoretical interactions or contradict widely accepted facts about code smell co-occurrence.

Step 5. Findings and Trustworthiness Assessment - Based on the contrastive analysis, inconsistencies and discrepancies were identified. If the dataset exhibited relationships that strongly contradicted known facts, its trustworthiness was questioned. Additionally, datasets with weak or non-existent interactions where strong interactions were expected were flagged as potentially unreliable. The findings provided insights into the quality and reliability of these datasets, offering recommendations for their improvement or cautious usage in further research.

III. Dataset Selection and Pre-processing

This section talks about the selection of datasets used for experimentation. It also discusses the creation and chrematistics of multilabel datasets. This section is divided into two sub-sections. The first sub-section A discusses the Dataset Group A. Another sub-section B talks about the Dataset Group B.

Since its debut, the Java programming language has seen a sharp increase in popularity, partly because of the Internet's ubiquitous use. According to the literature [18] [19][20], Java is thought to be the programming language that has been investigated the most in terms of code smells.

A. Group A Dataset

The study in this paper has used 4 notable published datasets (built over open-source Java Projects) by Arcelli Fontana et al. [16] for 4 different code smells: long method, feature envy, data class, and god class because these datasets are manually validated ones and have been used in the research extensively[20]. Hence, they were judged appropriate for the experimentation. To create the dataset, the authors [16] selected 74 heterogeneous Java open-source systems that belonged to diverse application domains from *Qualitas Corpus* [21]. The various project metrics at class and method level were computed using the tool *DFMC4J* (*Design Features and Metrics for Java*). Afterward, the authors labeled code smells to projects using code smell detection tools (*iPlasma*), *PMD1, Fluid Tool*, *AntiPattern Scanner,* and rules, followed by a manual validation done by 3 trained students for the specified task. All these students first performed individual evaluations of code smells and then they discussed among themselves and came to a consensus. Their discussion resulted in a set of guidelines for identifying each reported code smell. Thus, a total of four datasets were generated, with each dataset representing one of the four types of code smells: long method, feature envy, data class, and God class. A listing of features and the definitions of the features are made available in the appendix of her paper [16]. Each dataset contained one-third of smelly samples and two-thirds of non-smelly samples. Table I shows the characteristics of the datasets.

Since the 4 datasets as explained in Table I contains individual code smells but for smell interaction it requires the knowledge of samples which contain both code smells. Therefore, the study in this paper has used the multilabel dataset for long method and feature envy available in the paper [22].

This multilabel dataset has been downloaded from the link. Since the multilabel dataset for God class and data class was not available so a multilabel dataset for God class and data class was created from the individual datasets of God class i.e. $Dataset_2$ and data class (explained in Table III) i.e. $Dataset_1$ using the approach given by [22].

TABLE I
Code Smells Datasets

SI No.	Dataset name	Code smell	Number of features	No. of smelly samples	No. of –non-smelly samples
1	$Dataset_1$	DC	62	140	280
2	$Dataset_2$	GC			
3	$Dataset_3$	LM	82		
4	$Dataset_4$	FE			

Through observation, it was found that there were 393 samples common between the God Class and Data Class datasets. Additionally, 27 samples from the God Class dataset did not match any samples from the Data Class dataset, and similarly, 27 samples from the Data Class dataset did not match any samples from the God Class dataset. After merging the God Class and Data Class datasets to create a multilabel dataset, a total of 447 samples were available in the resulting multilabel dataset. The characteristics of both multilabel datasets are presented in Tables II-III. The multilabel dataset for GC and DC smells consists of 447 samples. Notably, no samples (0%) are affected by both GC and DC smells simultaneously, indicating no overlap between these two categories.

TABLE II
Datasets Characteristics of Multilabel Dataset for GC and DC Smells

No of samples	No of samples affected by both GC and DC (%)	No of samples affected by DC but not by GC (%)	No of samples affected by GC but not by DC (%)	No of samples not affected by both (%)
447	0 (0%)	140 (31%)	140 (31%)	167 (37%)

Additionally, 140 samples (31%) are affected solely by DC and not by GC, while another 140 samples (31%) are impacted only by GC and not by DC. Lastly, 167 samples (37%) are unaffected by either GC or DC smells, making them the largest group in the dataset.

TABLE III
Characteristics of Multilabel Dataset for LM and FE Smells

Total no of samples	No of samples affected by both (%)	No of samples affected by LM but not by FE (%)	No of samples affected by FE but not by Long method (%)	No of samples not affected by both (%)
445	100 (22.4%)	62 (13.9%)	40 (8.9%)	243 (54.6%)

The multilabel dataset for LM and FE smells consists of 445 samples, a notable portion of the dataset, 100 samples (22.4%), exhibit both LM and FE smells, indicating some overlap between the two smell types. Additionally, 62 samples (13.9%) are affected exclusively by LM but not by FE, and 40 samples (8.9%) are affected solely by FE. The majority of the dataset, 243 samples (54.6%), remains unaffected by either smell.

B. Group B Dataset

This section talks about the details of second set of experimental datasets. For this, the study has used an industry-relevant dataset of code smells.

Initially, Madeyski et al. [13] developed an industry-relevant dataset called "MLCQ," which focuses on four code smells: Feature Envy (FE), Long Method (LM), Data Class (DC), and Blob (BB). This dataset, compiled from current Java open-source projects, involved input from 26 professional software developers and includes nearly 15,000 code samples. The MLCQ dataset specifies the type and severity of each code smell, as well as its location in the source code, though it lacks detailed source code metrics.

Subsequently, Madeyski et al. [17] utilized the PMD and CODEBEAT platforms to compute various metrics. PMD is an open-source tool designed to analyze source code and calculate metrics. Although its default metrics are limited, PMD allows for the creation of custom user-defined metrics and is frequently employed by researchers for generating code metrics. CODEBEAT, on the other hand, provides metrics generated by an industrial partner and has not yet been extensively used in research. CODEBEAT's metrics span multiple programming languages and were assessed for their utility in code smell detection as part of expanding their product's capabilities.

As detailed in [17], the datasets for our research was obtained from https://doi.org/10.5281/zenodo.7319860. This dataset contains more than 14,000 reviews of over 5,000 Java code samples that were examined by 26 software developers. Reviews are specified for four code smells—Blob (BB), Data Class (DC), Feature Envy (FE), and Long Method (LM). There are four different severity levels of a review for a code sample: none(0), minor(1), major(2), and critical(3). Table IV offers a comprehensive analysis of the quantity of samples, characteristics, and level of severity assigned to every code sample.

The data in Table IV reveals a significant imbalance, with a large number of non-smelly samples compared to smelly ones when considering samples tagged with the severity label 'none' as non-smelly, while those labelled as 'minor,' 'major,' or 'critical' as smelly. Given the need for binary classification in our experiment, we employed methods to construct the datasets, which are discussed in the following section B.1.

TABLE IV
CHARACTERISTICS OF DOWNLOADED DATASETS

S. NO.	Smell	Total no. of samples	No. of '*critical*' samples	No. of '*major*' samples	No. of '*minor*' samples	No. of '*none*' samples	No. of features
1	LM	3362	78	274	454	2556	20
2	FE	3337	24	142	288	2883	20
3	DC	4021	146	401	510	2964	51
4	BB	4019	127	312	535	3045	51

B.1 Construction of experimental datasets

The proposed method of construction for multilabel datasets contains three steps which are as follows:

Step 1. (Calculation of binary labels): Step 1 calculates the binary labels (smelly (0), non-smelly (1)) of each sample in each dataset.

In the four downloaded datasets, there were multiple reviews (assigned by a reviewer) for the same code sample for some code samples. However, the datasets used for experimentation using ML needed a single severity label (smelly or non-smelly) for each code sample due to a binary classification problem, so to handle this, our study have used 4 techniques: a majority voting approach, mean approach, median and median_2 approach. Each method captures a different interpretation of reviewer consensus: majority voting reflects collective agreement, mean captures average severity, median provides a robust central tendency, and median_2 ensures high confidence labelling by focusing only on critical cases. By employing this multi-faceted aggregation strategy, we aimed to: Understand the effect of severity interpretation on dataset consistency; Ensure robustness of results across labelling assumptions. These techniques align with standard practices in areas like crowdsourced annotation, medical diagnosis aggregation, and subjective quality assessment, where multiple reviewers may disagree and label aggregation becomes critical for downstream tasks.

Step 2. (Filtering of samples): Step 2 filters the samples having some missing values marked with 'NA' in the obtained datasets.

Since some dataset samples had missing feature values (represented as "NA"), we had two options: (1) impute the missing values using a suitable technique, or (2) discard the affected samples. We chose the second option—discarding these samples—because retaining and imputing the missing values could compromise the overall accuracy of the dataset. Additionally, since LM and FE are method-level smells, and DC and BB are class-level smells, certain feature columns were not applicable to them. Consequently, these columns were removed from the constructed datasets because they contained 'NA' values in every row.

Step 3. (Creation of multilabel datasets): Step 3 is used to create the multilabel datasets for LM and FE as well as DC and BB smells as LM and FE occur at method level while DC and BB occurs at class-level. Since the objective was to determine which samples were affected by both smells, by only one, or by none, the study utilized an outer join to merge the two datasets, LM and FE, based on the common sample_id column. This approach ensured that all samples from both datasets were included in the multilabel result, even if they appeared in only one dataset. After merging, a classification column was added to categorize each sample: those with non-null values in both

datasets were labelled as "Affected by both," those with values in only one dataset were labelled as "Affected by LM only" or "Affected by FE only," and those with null values in both were labelled as "Not affected." This methodology provided a comprehensive view of the samples' statuses and facilitated a detailed analysis of the smells' impact. The similar process was followed to merge datasets of DC and BB smells.

B.1.1 Majority voting approach

Majority voting is specifically useful when the data is subjective or prone to errors, such as user-generated content or expert opinions. In such cases, majority voting can help reduce the impact of outlier or erroneous assessments by relying on the most common label. Thus, in our approach, if the majority of reviews are 'none' we assign a severity level of zero (0) i.e. non-smelly otherwise we assign a severity label of one (1) i.e. smelly. Thus, we have constructed the datasets using the pseudocode given in Appendix A.

Table V shows the characteristics of constructed datasets using majority voting. LM has the highest number of positive samples (265), while FE has the fewest (106). DC and BB have higher percentages of positive samples (26% and 21%, respectively) compared to LM (12%) and FE (5%). The DC and BB smells have more features (47) compared to LM and FE (15), which might indicate more complexity in the feature set for these smells.

TABLE V
CHARACTERISTICS OF CONSTRUCTED DATASETS BY MAJORITY VOTING

S. NO.	Smell	Total no of samples	No of +ve samples (smelly)	No of -ve Samples (non-smelly)	No of features	% of +ve samples	% of -ve samples
1	LM	2147	265	1882	15	12%	88%
2	FE	2134	106	2028	15	5%	95%
3	DC	1051	277	774	47	26%	74%
4	BB	1049	216	833	47	21%	79%

Overall, the data suggests varying distributions of code smells and their complexity, with some smells showing a higher reduction in sample size and others having a higher percentage of positive samples. The number of features also varies significantly, which could impact the effectiveness of detection techniques.

Table VI shows the characteristics of multilabel datasets of LM and FE. The multilabel dataset contains 2134 samples. 202 samples are affected by LM issue only, which is approximately 9.47% of the total samples. 43 samples are affected by the Field Error (FE) issue only, which is about 2.02% of the total samples.

63 samples are affected by both LM and FE, which makes up around 2.95% of the total samples. 1826 samples are unaffected by either issue, constituting 85.59% of the dataset. A small percentage (9.47%) is affected only by LM, and an even smaller percentage (2.02%) is affected only by FE. The overlap between LM and FE (i.e., samples affected by both issues) is minimal at 2.95%.

TABLE VI
CHARACTERISTICS OF MULTILABEL DATASETS (LM AND FE)

S. NO.	Multilabel Dataset	No. of samples affected by LM only	No. of samples affected by FE only	No. of samples affected by LM and FE both	No. of samples affected by none
1.	LM and FE	202	43	63	1826

TABLE VII
CHARACTERISTICS OF MULTILABEL DATASETS (BB AND DC)

S. NO.	Multilabel Dataset	No. of samples affected by DC only	No. of samples affected by BB only	No. of samples affected by DC and BB both	No. of samples affected by none
1.	DC and BB	224	163	53	609

Similarly, table VII shows that characteristics of multilabel datasets of DC and BB. The multilabel dataset contains 1051 samples. 224 samples are affected by DC (Design Complexity) only, which constitutes 21.31% of the total samples. 163 samples are affected by BB (Bug Burden) only, making up 15.51% of the total samples. 53 samples are affected by both DC and BB, representing 5.04% of the total samples. 609 samples are unaffected by either issue, comprising 57.94% of the dataset. A significant portion (57.94%) of the dataset is unaffected by either DC or BB. The highest percentage (21.31%) of affected samples comes from those impacted only by DC, followed by 15.51% affected only by BB. A relatively small percentage (5.04%) is affected by both

B.1.2 The mean approach

The mean approach is used because it provides a simple, balanced, and consistent way to aggregate review severity, making classification easier and more efficient. The mean approach calculates the average (mean) value of a set of numbers to represent the central tendency of the data. In the context of reviews or ratings, it involves summing all the ratings for a sample and dividing by the number of reviews to get an overall score. This score is then used for classification or decision-making. The pseudocode provided outlines a process for classifying a set of samples X= {x1, x2…, xn} based on their review severity levels. Each sample xi has a set of reviews Ri= {ri1, ri2…, rim}, where each review can have a severity value ranging from 0 to 3. The algorithm calculates the mean severity of the reviews for each sample. If the mean severity exceeds 1.5, the sample is classified as "smelly" (label 1), otherwise, it is classified as "non-smelly" (label 0). The labels are collected in a list and returned at the end. The key steps include calculating the mean severity of reviews and using it to determine the classification of each sample. The pseudocode for the same is given in Appendix B.

TABLE VIII
CHARACTERISTICS OF CONSTRUCTED DATASETS BY MEAN APPROACH

S. NO.	Smell	Total no of samples	No of +ve samples (smelly)	No of -ve Samples (non-smelly)	No of features	% of +ve samples	% of -ve samples
1	LM	2147	108	2039	15	5.03	94.97
2	FE	2134	12	2122	15	0.56	99.44
3	DC	1051	102	949	47	9.70	90.3
4	BB	1049	62	987	47	5.91	94.09

The multilabel dataset in Table IX contains 2135 samples, out of which 104 samples, or 4.8%, are affected only by LM, and 8 samples, representing 0.3%, are affected only by FE. A very small portion, 4 samples (0.1%), are impacted by both LM and FE. Meanwhile, 2018 samples, which account for 94.5% of the dataset, are unaffected by either issue.

TABLE IX
CHARACTERISTICS OF MULTILABEL DATASET (LM AND FE)

S. NO.	Multilabel Dataset	No. of samples affected by LM only	No. of samples affected by FE only	No. of samples affected by LM and FE both	No. of samples affected by none
1.	LM and FE	104	8	4	2018

The majority of the dataset remains unaffected, with the largest affected group being those impacted by both LM and FE.

The multilabel dataset in Table X analysis reveals that out of 1049 total samples, 101 are affected by DC only, while 61 samples are impacted by BB only. A minimal number, just 1 sample, is affected by both DC and BB, indicating that the two issues rarely occur together. Additionally, 886 samples are unaffected, comprising about 84.5% of the dataset, which suggests that the majority of the samples are in good condition. This highlights that Data Corruption has a greater impact compared to Bad Blocks, and the co-occurrence of both problems is uncommon, showing that the dataset is relatively stable with a large proportion of unaffected samples.

TABLE X
CHARACTERISTICS OF MULTILABEL DATASET (BB AND DC)

S. NO	Multilabel Dataset	No. of samples affected by DC only	No. of samples affected by BB only	No. of samples affected by DC and BB both	No. of samples affected by none
1.	DC and BB	101	61	01	886

B.1.3 The median approach

The median approach is useful when the data contains outliers or is skewed, as it is less affected by extreme values. It provides a more robust representation of the central tendency, especially for ordinal data. The median approach refers to using the middle value in a sorted list of numbers to represent the central tendency of the data. In the context of reviews or ratings, it involves sorting the ratings for a sample and selecting the middle value (or the average of the two middle values if there is an even number of reviews). This value is then used for classification or decision-making. The pseudocode for the median approach is given in Appendix C.

Table XI shows the characteristics of constructed datasets using the median approach.

The multilabel dataset in Table XII contains 2133 samples, out of which 214 samples, or 10%, are affected only by LM, and 25 samples, representing 1.1%, are affected only by FE. A smaller portion, 46 samples (2.1%), are impacted by both LM and FE. Meanwhile, 1849 samples, which account for 86.68% of the dataset, are unaffected by either issue. The majority of the dataset remains unaffected, with the largest affected group being those impacted by LM only.

TABLE XI
CHARACTERISTICS OF CONSTRUCTED DATASETS BY MEDIAN APPROACH

S. NO	Smell	Total no of samples	No of +ve samples (smelly)	No of -ve Samples (non-smelly)	No of features	% of +ve samples	% of -ve samples
1	LM	2147	260	1887	15	12.10	87.9
2	FE	2134	71	2063	15	3.33	96.67
3	DC	1051	253	798	47	24.07	75.93
4	BB	1049	178	871	47	16.97	83.03

The multilabel dataset in Table XIII shows that while both DC and BB are significant issues, DC only affects more samples than BB only, and the overlap between the two issues is relatively small. Still, a substantial portion of the samples (648, or 61.9%) remain unaffected, indicating that most of the dataset is free from these issues. This suggests that while DC and BB are present, they don't overwhelmingly affect the entire dataset.

TABLE XII
CHARACTERISTICS OF MULTILABEL DATASET (LM AND FE)

S. NO.	Multilabel Dataset	No. of samples affected by LM only	No. of samples affected by FE only	No. of samples affected by LM and FE both	No. of samples affected by none
1.	LM and FE	214	25	46	1849

TABLE XIII
CHARACTERISTICS OF MULTILABEL DATASET (DC AND BB)

S. NO.	Multilabel Dataset	No. of samples affected by DC only	No. of samples affected by BB only	No. of samples affected by DC and BB both	No. of samples affected by none
1.	DC and BB	223	148	30	648

B.1.4 The median_2 approach

This approach involves assigning binary labels to code smell samples based on the severity levels of multiple developer reviews. Each sample consists of several reviews, where the severity of a review can be categorized as None (0), Minor (1), Major (2), or Critical (3). To determine whether a sample is smelly (1) or non-smelly (0), the median severity level of all reviews for the sample is calculated. If the median severity is exactly "Critical" (3), the sample is marked as smelly (1). Otherwise, it is classified as non-smelly (0). This method ensures that only samples with a consistent consensus of critical severity across reviewers is flagged as smelly, making it more stringent and reducing false positives. Using the median as the aggregation method provides robustness against outlier reviews, ensuring the label reflects the central tendency of reviewer opinions. The pseudocode for the same is mentioned in Appendix D.

TABLE XIV
CHARACTERISTICS OF CONSTRUCTED DATASETS BY MEDIAN_2 APPROACH

S. NO	Smell	Total no of samples	No of +ve samples (smelly)	No of -ve Samples (non-smelly)	No of features	% of +ve samples	% of -ve samples
1	LM	2147	111	2036	15	5.17	94.83
2	FE	2134	15	2119	15	0.70	99.3
3	DC	1051	107	944	47	10.18	89.82
4	BB	1049	59	990	47	5.62	94.38

The multilabel dataset of LM and FE in Table XV contains 2135 samples, out of which 107 samples, or 5.01%, are affected only by LM, and 11 samples, representing 0.52%, are affected only by FE. A very small portion, 4 samples (0.19%), are impacted by both LM and FE.

Meanwhile, 2012 samples, which account for 94.28% of the dataset, are unaffected by either issue. The overwhelming majority of the dataset remains unaffected, with the largest affected group being those impacted by LM only.

TABLE XV
CHARACTERISTICS OF MULTILABEL DATASET (LM AND FE)

S. NO	Multi label Dataset	No. of samples affected by LM only	No. of samples affected by FE only	No. of samples affected by LM and FE both	No. of samples affected by none
1.	LM and FE	107	11	4	2012

In the multilabel dataset in Table XVI labelled "DC and BB", a total of 105 samples are affected by the "DC" smell exclusively, while 57 samples are impacted solely by the "BB" smell. Additionally, there are 2 samples that exhibit characteristics of both "DC" and "BB" smells.

Importantly, the majority of the samples, totalling 885, show no signs of being affected by either smell. This distribution highlights that while a small percentage of the dataset is impacted by specific code smells, the majority remains unaffected, indicating a potentially significant focus for further analysis or refactoring efforts. The relatively low overlap between the two types of smells suggests that they may represent distinct issues within the dataset.

TABLE XVI
CHARACTERISTICS OF MULTILABEL DATASET (DC AND BB)

S. NO.	Multilabel Dataset	No. of samples affected by DC only	No. of samples affected by BB only	No. of samples affected by DC and BB both	No. of samples affected by none
1.	DC and BB	105	57	2	885

B.1.5 Characteristics of multilabel datasets

Merging of datasets in this study serves several key purposes. It enables a comprehensive analysis of how different code smells, such as LM and FE, interact within the same samples, offering insights into their individual and combined impact. This approach helps detect overlaps, identifying samples affected by both smells, only one, or neither, which is essential for understanding potential interactions. Additionally, multilabel datasets allow for a consistent comparison across various methods (majority voting, mean, median, and median_2), revealing how each approach influences smell identification.

Four approaches—majority voting, mean, median, and median_2—were employed to generate distinct datasets for the LM, FE, BB, and DC code smells. For each approach, the study then multilabel the LM and FE datasets based on sample IDs. After the merge, the samples were categorized into those affected by only one smell (either LM or FE), by both smells, or by neither, providing a detailed understanding of the individual and combined impact of these smells. The same methodology was applied to the BB and DC datasets, where samples were similarly classified, ensuring a consistent framework for analysing smell overlaps and their effects across different datasets. The results of these analyses are presented below in Table XVII.

The Table XVII compares four methods—Majority voting approach, mean approach, Median approach, and Median_2 approach—based on the number of samples affected by LM (a specific smell), FE (another smell), both LM and FE, or neither. The Majority voting approach shows the highest number of samples affected by LM only (202) and by both LM and FE (63), but the lowest number of unaffected samples (1826). In contrast, the Mean approach significantly reduces the number of samples affected by both smells (4) and none (2018), indicating a more conservative detection.

The Median approach affects more samples than the Mean approach across all categories but remains similar in detecting unaffected samples. Lastly, the Median_2 approach, like the Mean approach, shows a low number of samples affected by both smells (4) and keeps a high count of unaffected samples (2012), reflecting its closeness to the Mean approach in behaviour.

TABLE XVII
CHARACTERISTICS OF MULTILABEL DATASETS (LM AND FE)

S. NO.	Method	No. of samples affected by LM only	No. of samples affected by FE only	No. of samples affected by LM and FE both	No. of samples affected by none
1.	Majority voting approach	202	43	63	1826
2	Mean approach	104	8	4	2018
3	Median approach	214	25	46	1849
4	Median_2 approach	107	11	4	2012

Overall, the Majority voting method detects the highest number of smelly samples, while the Mean and Median_2 methods are more conservative in identifying smelly cases.

TABLE XVIII
CHARACTERISTICS OF MULTILABEL DATASETS (BB AND DC)

S. NO.	Method	No. of samples affected by DC only	No. of samples affected by BB only	No. of samples affected by DC and BB both	No. of samples affected by none
1.	Majority voting approach	224	163	53	609
2	Mean approach	101	61	01	886
3	Median approach	223	148	30	648
4	Median_2 approach	105	57	2	885

The Table XVIII compares four methods—Majority voting approach, mean approach, Median approach, and Median_2 approach—based on the number of samples affected by DC (a specific smell), BB (another smell), both DC and BB, or none. The Majority voting approach detects the highest number of samples affected by DC only (224) and by both DC and BB (53) but has the lowest count of unaffected samples (609). The Mean approach is more conservative, with the fewest samples affected by DC and BB (1) and the highest number of unaffected samples (886). The Median approach closely mirrors the Majority voting approach, with high detection rates for DC and BB individually, but a lower number of samples affected by both (30) and a slightly higher count of unaffected samples (648). Finally, the Median_2 approach behaves similarly to the Mean approach, showing low detection for both DC and BB together (2) and a high number of unaffected samples (885). Overall, the Majority voting method identifies the highest number of smelly samples, while the Mean and Median_2 approaches are more conservative, identifying fewer affected samples.

IV. ANALYSIS OF MULTILABEL DATASETS FOR SMELL INTERACTION ANALYSIS

This section examines multilabel datasets with a specific focus on analysing code smell interactions. To conduct this analysis, the study utilizes two key metrics—confidence and lift—commonly employed in association mining. These metrics help assess the strength and significance of relationships between different code smells, providing valuable insights into their co-occurrence patterns. This section is divided into 2 sub-sections A and B. The sub-section A provides the analysis of first set of multilabel datasets and section B discusses the analysis of second set of multilabel datasets.

Confidence and lift are key metrics in association rule mining, a technique used to discover relationships (associations) between items in large datasets. Confidence shows how strong a rule is, while lift tells us if the relationship is significant or merely a coincidence. Confidence reflects the strength of prediction based on the antecedent, while lift indicates the strength of the association relative to chance. Both are crucial in analysing data patterns and determining useful relationships for decision-making.

Confidence is a measure in association rule mining that indicates how often the rule holds true. It shows the probability that the consequent(B) of a rule will occur given that the antecedent(A) has occurred. Confidence is calculated as the ratio of the number of transactions containing both the antecedent and the consequent, to the number of transactions containing only the antecedent.

Lift measures how much more likely the consequent (B) is to occur when the antecedent (A) occurs, compared to when B occurs independently of A. Lift evaluates whether the association between A and B is significant beyond chance. They are defined as follows:

> Confidence (A -> B) = (Support of A and B) / (Support of A)
> Lift (A -> B) = (Support of A and B) / (Support of A * Support of B)
> Where: A -> B is the association rule.
> "Support of B" is the frequency of B in the dataset. Support of A is the frequency of A in the dataset. "Support of A and B" is the frequency of both A and B occurring together.

If the confidence of the rule A -> B is 80%, it means that 80% of the transactions containing A also contain B. A higher confidence indicates a stronger association between A and B, but it doesn't consider how common B is in the dataset. When the lift value is equal to 1, it indicates that A and B are independent, meaning there is no association between them—they occur together purely by chance. If the lift value is greater than 1, it signifies a positive correlation, where the presence of A increases the likelihood of B occurring. Conversely, a lift value less than 1 suggests a negative correlation, where the presence of A decreases the likelihood of B occurring. For example, if the lift for the rule A -> B is 1.5, it means A makes B 1.5 times more likely to occur compared to when B occurs on its own.

A. Dataset Group A (Fontana Dataset)

This section analyses the multilabel datasets for the Dataset Group A, specifically the Fontana Dataset [16]. The characteristics of these multilabel datasets are summarized in Tables VI and VII. Furthermore, Tables XIX and XX present the computed confidence and lift values derived from the data in Table XVIII.

TABLE XIX
CONFIDENCE SCORES

S. No.	Confidence (LM to FE)	Confidence (FE to LM)	Confidence (DC to GC)	Confidence (GC to DC)
1	61.7%	71.4%	0%	0%

The confidence for LM to FE is 61.7%, indicating that when a Long Method smell occurs, there is a 61.7% chance that Feature Envy will also be present. Similarly, the confidence for FE to LM is slightly higher at 71.4%, suggesting that the occurrence of Feature Envy has a 71.4% probability of also having a Long Method. However, the confidence values for both DC to GC and GC to DC are 0%, implying no association between Data Class and God Class smells in this scenario. Overall, the data shows a strong bidirectional association between LM and FE but no detectable relationship between DC and GC.

TABLE XX
LIFT SCORES

S. No.	Lift (LM to FE)	Lift (DC to GC)
1	1.9 (+ve)	0

The lift for LM to FE is 1.9, indicating a strong positive correlation, meaning the occurrence of a Long Method smell makes Feature Envy 1.9 times more likely than if they were independent. On the other hand, the lift for DC to GC is 0, signifying no association between Data Class and God Class smells in this context, implying that the presence of a Data Class smell does not influence the occurrence of a God Class smell. Overall, there is a notable association between LM and FE, while DC and GC show no detectable relationship.

B. Dataset Group B (Industry Relevant datasets)

This section presents the analysis of multilabel datasets for the Dataset Group B namely Industry relevant datasets [17]. The characteristics of the corresponding multilabel datasets are detailed in Table XVIII. Additionally, Tables XXI and XXII provide the computed values of confidence and lift based on the data from Table XVIII.

The data in Table XXI compares four approaches—Majority voting, Mean, Median, and Median_2—based on their confidence values for various code smell associations: Long Method (LM) to Feature Envy (FE), FE to LM, Data Class (DC) to God Class (GC), and GC to DC. The Majority voting approach shows relatively high confidence, particularly in the FE to LM (146.51%) and GC to DC (32.52%) associations, but with a lower confidence for LM to FE (31.19%) and DC to GC (23.66%).

The Mean approach consistently has the lowest confidence across all associations, with very minimal values, particularly DC to GC (0.99%) and GC to DC (1.64%). The Median approach displays stronger confidence in FE to LM (184.00%) and reasonable values in the other associations, with moderate confidence in LM to FE (21.50%). The Median_2 approach shows a more balanced, though lower, performance compared to the Median approach, with confidence values of 36.36% (FE to LM) and 3.51% (GC to DC). Overall, the Majority voting and Median approaches outperform the others in detecting these code smell associations.

TABLE XXI
CONFIDENCE SCORES

S. No	Approach	Confidence (LM to FE)	Confidence (FE to LM)	Confidence (DC to GC)	Confidence (GC to DC)
1	Majority voting approach	31.19%	146.51%	23.66%	32.52%
2	Mean approach	3.85%	50.00%	0.99%	1.64%
3	Median approach	21.50%	184.00%	13.45%	20.27%
4	Median_2 approach	3.74%	36.36%	1.90%	3.51%

The analysis of lift values shown in Table XXII across various approaches—Majority Voting, Mean, Median, and Median_2—reveals a consistent negative correlation between Long Method (LM) and Feature Envy (FE), as well as between Data Class (DC) and God Class (GC). All approaches yield lift values below 1, indicating that the presence of one code smell reduces the likelihood of the other occurring.

Specifically, the Majority Voting and Median approaches demonstrate relatively higher lift values for LM to FE (0.7253 and 0.8598, respectively), suggesting a lesser decrease in likelihood compared to the Mean and Median_2 approaches, which show significantly lower values.

The association between DC and GC is notably weak across all methods, with lift values as low as 0.0162 in the Mean approach, indicating minimal interaction. The negative lift values for both LM to FE and DC to GC suggest that these pairs of code smells do not co-occur frequently. This implies that the underlying problems or structural issues they represent are likely distinct from each other.

For the "DC to GC" association, the Mean approach gives a lift value of 0.0162, which is very close to zero, indicating that DC and GC almost never co-occur. This is a much stronger negative association than, say, the Median approach's lift value of 0.0909, which suggests a slightly weaker negative relationship.

TABLE XXII
LIFT SCORES

S. No.	Approach	Lift (LM to FE)	Lift (DC to GC)
1	Majority voting approach	0.7253 (-ve)	0.1452 (-ve)
2	Mean approach	0.4808 (-ve)	0.0162 (-ve)
3	Median approach	0.8598 (-ve)	0.0909 (-ve)
4	Median_2 approach	0.3398 (-ve)	0.0334 (-ve)

C. Contrasting with Established Facts

This section is divided into three sub-sections. Sub-section C.1 explores established findings from the literature regarding

the relationship between LM and FE. Sub-section C.2 examines existing literature on the relationship between DC and GC. Finally, Sub-section C.3 compares the findings from C.1 and C.2 with the results obtained in Section 4, providing key insights and analysis.

C.1 Relationship between LM and FE

According to Martin fowler et al. [1] Long method and Feature envy has been defined as follows: FE: A classic smell is a method that seems more interested in a class other than the one it actually is in. LM: Almost all the time the problems come from methods that are too long. Long methods are troublesome because they often contain lots of information, which gets buried by the complex logic that usually gets dragged in. The relationship between LM and FE has been a central focus in the study of code smells. These two smells are often considered interrelated due to their impact on code modularity and cohesion.

Lozano [23] analyzed 3 different open-source Java projects: 16 releases of Log4j from version 1.2.1 to 1.2.17, 34 releases of Jmol from versions 1 until 11.0 and 45 releases of JFreeChart (from version 0.5.6 to 1.0.14). Lozano [23] discovered a strong correlation between LM and FE and found that almost all methods with FE were in LMs, which means FE methods are included in LMs. They found an inclusion relation between FE and LM i.e., whenever a FE method(s) exists then it is included in a LM. In other words, FE is accompanied by LM. Palomba [24] analyzed 395 releases of 30 open-source Java projects and found co-occurrence of LM and FE code smells. They explained that as LMs are made up of numerous code statements and naturally depend on other classes, they are more likely to be influenced by the FE code smell. They found that support (LM, FE) = 0.91 and confidence (LM, FE) = 0.99. Later on, Fontana [25] also found a strong co- occurrence of LM and FE code smells. Similar findings are echoed in other studies, such as those by Mäntylä et al. [26], which emphasize the high co-occurrence of LM and FE. In their investigation, Feature Envy is frequently observed in conjunction with overly complex methods. This suggests that the structural weaknesses exposed by Long Method, such as sprawling codebases, could contribute to methods accessing foreign classes inappropriately, thus leading to Feature Envy.

C.2 Relationship between GC and DC

According to Martin fowler et al. [1] Data class and Large class has been defined as follows: Data class: These are classes that have fields, getting and setting methods for the fields, and nothing else. Such classes are dumb data holders and are almost certainly being manipulated in far too much detail by other classes. Large class/God Class: When a class is trying to do too much, it often shows up as too many instance variables. When a class has too many instance variables, duplicated code cannot be far behind. The relationship between God Class (GC) and Data Class (DC) is notably different.

Paper [27] found a used relation between GC and DC i.e., in some cases, when data classes were utilized by other classes, GC affects most of them. However, they did not report any instances of GC and DC code smell co-occurring. The co-occurrence of GC and DC has not been seen in recent research [20] on code smell relations. Yamashita [28] earlier found that god classes may be coupled with data classes. We also observed that Rasool et al. [29] kept GC and DC into different groups: Mass-based categorization and Feature-based categorization respectively. Code smells grouped using mass-based categorization were based on the size of the objects, methods, and arguments. Feature-based categorization, on the other hand, contained code smells according to incomplete, improper, or missing behavior. Earlier Mantyla [30] has also kept god class and data class in separate categories: Bloater and Dispensables. Bloaters are elements in the code that have grown too big to manage properly, whereas Dispensables are elements that are superfluous and should be deleted. A study by Lanza and Marinescu [31] further highlights the differences between these smells. God Class tends to emerge from an accumulation of responsibilities, whereas Data Class results from poor design where classes serve as mere data containers. Their divergent roles in code architecture mean that they do not typically influence each other's occurrence. However, some studies, like those by Khomh et al. [15], have pointed out that these smells may indirectly affect code quality together. For example, a God Class might arise when too much data is consolidated from multiple Data Classes. This suggests that while direct correlations are rare, indirect architectural dependencies could exist, particularly in large legacy systems. The study provides further evidence that the relationship between these smells is context-dependent, often varying across software systems depending on project size and architecture.

C.3 Comparison with Experimental Findings and Insights

According to Martin Fowler's definitions of GC and DC, these two code smells cannot co-occur—GC involves excessive responsibilities, whereas DC lacks any functionality. Analysing Table XIX and Table XX (for dataset Group A), it is evident that GC and DC have no significant relationship, meaning the presence of one does not influence the other. Additionally, Table II confirms that no sample is affected by both GC and DC simultaneously. These observations align with the established facts outlined in Section C.2. Moreover, the strong confidence and lift values in Table XIX and Table XX for LM and FE suggest a bidirectional association between these two code smells. Specifically, the presence of LM increases the likelihood of FE, which is consistent with the insights discussed in Section C.1.

However, for dataset Group B, Table XVIII reveals that GC and DC occur only negligibly in the multilabel datasets constructed using the median_2 and mean approaches. This observation is further supported by the low lift scores for GC and DC in Table XXII, indicating that their co-occurrence is highly unlikely. The DC to GC relationship remains consistent with past studies, affirming that these two smells are not significantly related. Additionally, Table XXI reports very low confidence values for both GC to DC and DC to GC, reinforcing the weak or non-existent association between these two code smells. Furthermore, Table XXII shows that the lift scores for GC and DC are significantly lower across all approaches compared to those for LM and FE. This further confirms that the likelihood of GC and DC co-occurring is much lower than that of LM and FE. However, in this dataset, all approaches show negative lift values (e.g., Majority Voting: -0.7253, Mean:

-0.4808, etc.), suggesting an absence of a meaningful relationship. The negative lift values for LM to FE in Table XXII contradict prior research, which found strong support and confidence values. This raises concerns about potential inconsistencies in the dataset or the effectiveness of the applied approaches. These findings suggest that the dataset does not capture the expected dependencies between LM and FE, warranting a re-evaluation of detection techniques or dataset quality. Therefore, we believe that this is the primary reason for the poor machine learning results reported in paper [17] on Group B datasets. These deviations likely arise from labelling inconsistencies or reviewer bias, underscoring the need to validate industrial datasets before ML use.

V. Conclusion & Future Work

The study introduces a novel approach to assessing the validity and reliability of code smell datasets. For experimentation, two widely recognized and manually validated sets of publicly available datasets (categorized as Group A and Group B) were used and were analysed using confidence and lift, two key metrics in association rule mining, to evaluate the degree of interaction among code smells. The findings were then compared against established facts from the literature. The results indicate that the Group A dataset aligns well with known facts, whereas the Group B dataset exhibits some deviations from the expected patterns. This raises concerns about the efficacy of the used techniques or possible discrepancies in the Group B dataset. Differences may be attributed to contextual factors such as domain-driven requirements, project-specific coding cultures, or changes in prevailing software engineering paradigms. However, there are notable differences between Group A and Group B datasets. Group A datasets are more balanced compared to Group B but are smaller in size. Additionally, Group A datasets cannot be considered industry relevant. This highlights the critical need for a large-scale dataset in the context of code smell detection that aligns with established knowledge. Observed deviations should be interpreted cautiously as potential indicators of dataset noise, mislabelling, or contextual differences in industrial vs. academic coding practices, rather than evidence against established smell relations. Researchers can apply our framework to assess dataset consistency before using them in ML models, ensuring more reliable results. It is recommended that future datasets be accompanied by quality indicators, such as consistency metrics, to support the assessment of their reliability. Furthermore, it is suggested that detection tools be adapted to the characteristics of the dataset (e.g., academic vs. industrial) to enhance their effectiveness in real-world applications. Open collaboration and transparency in dataset construction will further enhance reproducibility, strengthening the foundation for more effective code smell detection and software quality assessment.

While statistical tests such as permutation or bootstrapping were not applied in this study, their use is acknowledged as valuable for validating association patterns. It is suggested that such methods be incorporated in future work to enhance the reliability of the findings. Advanced evaluation metrics, such as causal inference can enhance code smell analysis. To improve dataset reliability, researchers should ensure cross-validation with empirical data, standardize labelling practices, involve domain experts, and periodically reassess datasets against evolving coding standards.

References

[1] B. Martin Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*. 2002. Accessed: May 02, 2019. [Online]. Available: https://www.csie.ntu.edu.tw/~r95004/Refactoring_improving_the_design_of_existing_code.pdf

[2] T. Hall, M. Zhang, D. Bowes, Y. S.-A. T. on Software, and undefined 2014, "Some code smells have a significant but small effect on faults," *dl.acm.orgT Hall, M Zhang, D Bowes, Y SunACM Trans. Softw. Eng. Methodol. (TOSEM), 2014*, vol. 9, p. 39, 2013, doi: 10.1145/0000000.0000000.

[3] F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, D. Poshyvanyk, and A. De Lucia, "Mining version histories for detecting code smells," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 462–489, 2015, doi: 10.1109/TSE.2014.2372760.

[4] B. Pietrzak and B. Walter, "Leveraging code smell detection with inter-smell relations," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4044 LNCS, no. June 2006, pp. 75–84, 2006, doi: 10.1007/11774129_8.

[5] A. Yamashita and L. Moonen, "Do code smells reflect important maintainability aspects?," *IEEE Int. Conf. Softw. Maintenance, ICSM*, pp. 306–315, 2012, doi: 10.1109/ICSM.2012.6405287.

[6] 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," *Empir. Softw. Eng.*, vol. 23, no. 3, pp. 1188–1221, Jun. 2018, doi: 10.1007/s10664-017-9535-z.

[7] F. A. Fontana, J. Dietrich, B. Walter, A. Yamashita, and M. Zanoni, "Anti-pattern and code smell false positives: Preliminary conceptualisation and classification," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2016*, 2016, pp. 609–613. doi: 10.1109/SANER.2016.84.

[8] M. Tufano *et al.*, "When and Why Your Code Starts to Smell Bad (and Whether the Smells Go Away)," *IEEE Trans. Softw. Eng.*, vol. 43, no. 11, pp. 1063–1088, 2017, doi: 10.1109/TSE.2017.2653105.

[9] H. Nandani, M. Saad, and T. Sharma, "DACOS - A Manually Annotated Dataset of Code Smells," in *Proceedings - 2023 IEEE/ACM 20th International Conference on Mining Software Repositories, MSR 2023*, 2023, pp. 446–450. doi: 10.1109/MSR59073.2023.00067.

[10] R. Sandouka and H. Aljamaan, "Python code smells detection using conventional machine learning models," *PeerJ Comput. Sci.*, vol. 9, 2023, doi: 10.7717/peerj-cs.1370.

[11] F. Zhang *et al.*, "Data preparation for Deep Learning based Code Smell Detection: A systematic literature review," *J. Syst. Softw.*, vol. 216, 2024, doi: 10.1016/j.jss.2024.112131.

[12] M. Zakeri-Nasrabadi, S. Parsa, E. Esmaili, and F. Palomba, "A Systematic Literature Review on the Code Smells Datasets and Validation Mechanisms," *ACM Comput. Surv.*, vol. 55, no. 13 s, Dec. 2023, doi: 10.1145/3596908.

[13] L. Madeyski and T. Lewowski, "MLCQ: Industry-Relevant Code Smell Data Set," in *ACM International Conference Proceeding Series*, 2020, pp. 342–347. doi: 10.1145/3383219.3383264.

[14] E. D. S. Maldonado and E. Shihab, "Detecting and quantifying different types of self-admitted technical Debt," in *2015 IEEE 7th International Workshop on Managing Technical Debt, MTD 2015 - Proceedings*, 2015, pp. 9–15. doi: 10.1109/MTD.2015.7332619.

[15] Z. Soh, A. Yamashita, F. Khomh, and Y. G. Guéhéneuc, "Do code smells impact the effort of different maintenance programming activities?," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2016*, 2016, pp. 393–402. doi: 10.1109/SANER.2016.103.

[16] F. Arcelli Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empir. Softw. Eng.*, vol. 21, no. 3, pp. 1143–1191, 2016, doi: 10.1007/s10664-015-9378-4.

[17] L. Madeyski and T. Lewowski, "Detecting code smells using industry-relevant data," *Inf. Softw. Technol.*, vol. 155, 2023, doi: 10.1016/j.infsof.2022.107112.

[18] T. Sharma and D. Spinellis, "A survey on software smells," *J. Syst. Softw.*, vol. 138, no. December 2017, pp. 158–173, 2018, doi: 10.1016/j.jss.2017.12.034.

[19] A. AbuHassan, M. Alshayeb, and L. Ghouti, "Software smell detection

techniques: A systematic literature review," *J. Softw. Evol. Process*, no. September 2019, pp. 1–48, 2020, doi: 10.1002/smr.2320.
[20] T. Lewowski and L. Madeyski, "Code Smells Detection Using Artificial Intelligence Techniques: A Business-Driven Systematic Review," *Dev. Inf. Knowl. Manag. Bus. Appl.*, vol. 3, pp. 285–319, 2022, doi: 10.1007/978-3-030-77916-0_12.
[21] E. Tempero *et al.*, "The Qualitas Corpus: A curated collection of Java code for empirical studies," in *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, 2010, pp. 336–345. doi: 10.1109/APSEC.2010.46.
[22] T. Guggulothu and S. A. Moiz, "Code smell detection using multi-label classification approach," *Softw. Qual. J.*, 2020, doi: 10.1007/s11219-020-09498-y.
[23] A. Lozano, K. Mens, and J. Portugal, "Analyzing code evolution to uncover relations between bad smells," in *2015 IEEE 2nd International Workshop on Patterns Promotion and Anti-Patterns Prevention, PPAP 2015 - Proceedings*, 2015, pp. 1–4. doi: 10.1109/PPAP.2015.7076847.
[24] F. Palomba, R. Oliveto, and A. De Lucia, "Investigating code smell co-occurrences using association rule learning: A replicated study," *MaLTeSQuE 2017 - IEEE Int. Work. Mach. Learn. Tech. Softw. Qual. Eval. co-located with SANER 2017*, pp. 8–13, 2017, doi: 10.1109/MALTESQUE.2017.7882010.
[25] B. Walter, F. A. Fontana, and V. Ferme, "Code smells and their collocations: A large-scale experiment on open-source systems," *J. Syst. Softw.*, vol. 144, no. June, pp. 1–21, 2018, doi: 10.1016/j.jss.2018.05.057.
[26] M. V. Mäntylä and C. Lassenius, *Subjective evaluation of software evolvability using code smells: An empirical study*, vol. 11, no. 3. 2006. doi: 10.1007/s10664-006-9002-8.
[27] F. A. Fontana, V. Ferme, and M. Zanoni, "Towards Assessing Software Architecture Quality by Exploiting Code Smell Relations," *Proc. - 2nd Int. Work. Softw. Archit. Metrics, SAM 2015*, no. May, pp. 1–7, 2015, doi: 10.1109/SAM.2015.8.
[28] A. Yamashita and L. Moonen, "To what extent can maintenance problems be predicted by code smell detection? -An empirical study," *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2223–2242, 2013, doi: 10.1016/j.infsof.2013.08.002.
[29] G. Rasool and Z. Arshad, "A Lightweight Approach for Detection of Code Smells," *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 483–506, 2017, doi: 10.1007/s13369-016-2238-8.
[30] M. Mantyla, J. Vanhanen, and C. Lassenius, "A taxonomy and an initial empirical study of bad smells in code," in *ieeexplore.ieee.org*, 2004, pp. 381–384. doi: 10.1109/icsm.2003.1235447.
[31] M. Lanza and R. Marinescu, *Object-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. 2006. doi: 10.1007/3-540-39538-5.

Appendices:https://docs.google.com/document/d/1iwZlRDxMrxbyO7x0pfGrkIWkBiqbiW8G/edit?usp=sharing&ouid=102221464989892511063&rtpof=true&sd=true

Ruchin Gupta is an accomplished academic and researcher with over 23 years of experience, currently serving as an Assistant Professor at JIIT, Noida. He is an active member of several prestigious engineering societies and has contributed extensively to the field through his roles as a reviewer for top-tier journals and as a technical chair at international conferences. His research spans machine learning, code smell detection, and blockchain technology.

Jitendra Kumar Seth is working as Professor in KIET Group of Institutions Ghaziabad. He has completed PhD degree in CSE from Jaypee Institute of Information Technology, Noida in 2019. He obtained M.Tech. degree from Shobhit University Meerut in 2009 and B.Tech. degree from Radha Govind Engineering College Meerut in 2004. During his teaching career of more than 20 years he has published 24 research papers in reputed International Journals, Book Chapters and Conferences. His research area includes Cloud Computing, Big Data, Cyber Security, Machine Learning and Search Engines.

Anupama Sharma is a Professor in the Department of Information Technology at Ajay Kumar Garg Engineering College, Ghaziabad. She holds a B.E. in Computer Science and Engineering (CSE), an M.Tech in CSE, and a Ph.D. in CSE. Her Ph.D. research focused on Quality of Service (QoS)-based performance enhancement in ad-hoc routing protocols. With 22 years of academic experience, Dr. Sharma's research interests include QoS routing in wireless networks, database management systems, algorithms, etc.

Abhishek Goyal is working as an Associate Professor at KIET Group of Institutions, Delhi-NCR, Ghaziabad, India. He obtained his Doctorate in Computer Science in 2024. Bachelor of Technology & Master of Technology in 2003 and 2012 from Chaudhary Charan Singh University, Meerut and Amity University Uttar Pradesh, Noida respectively. His research interests include Cloud Computing, Scheduling and optimization and multi-criteria decision-making optimization. He has published a number of research papers in reputed international journals and conference.