Predicting Community Smells' Occurrence on Individual Developers by Sentiments

Zijie Huang[†], Zhiqing Shao[†], Guisheng Fan[†], Jianhua Gao[§], Ziyi Zhou[†], Kang Yang[†], Xingguang Yang[†]
[†]*Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China*
[§]*Department of Computer Science and Technology, Shanghai Normal University, Shanghai, China*
hzj@mail.ecust.edu.cn, {zshao, gsfan}@ecust.edu.cn,
jhgao@shnu.edu.cn, zhouziyi@mail.ecust.edu.cn, {15921709583, xingguang2955}@163.com

***Abstract*—Community smells appear in sub-optimal software development community structures, causing unforeseen additional project costs, e.g., lower productivity and more technical debt. Previous studies analyzed and predicted community smells in the granularity of community sub-groups using socio-technical factors. However, refactoring such smells requires the effort of developers individually. To eliminate them, supportive measures for every developer should be constructed according to their motifs and working states. Recent work revealed developers' personalities could influence community smells' variation, and their sentiments could impact productivity. Thus, sentiments could be evaluated to predict community smells' occurrence on them. To this aim, this paper builds a developer-oriented and sentiment-aware community smell prediction model considering 3 smells such as Organizational Silo, Lone Wolf, and Bottleneck. Furthermore, it also predicts if a developer quitted the community after being affected by any smell. The proposed model achieves cross- and within-project prediction F-Measure ranging from 76% to 93%. Research also reveals 6 sentimental features having stronger predictive power compared with activeness metrics. Imperative and indicative expressions, politeness, and several emotions are the most powerful predictors. Finally, we test statistically the mean and distribution of sentimental features. Based on our findings, we suggest developers should communicate in a straightforward and polite way.**

***Index Terms*—community smell, developer sentiment, social debt, open source system, empirical software engineering**

I. Introduction

Software engineering is a complex social activity that incorporates both communication and collaboration of developers and other stakeholders [1]. The social and socio-technical features of development communities, along with the purely technical ones [1], have already been applied to predict and analyze software reliability [2] and maintainability [3]. Inspired by the conception of code smell [4], Palomba *et al.* [1] coined the term "community smell" to describe unhealthy organizational structure of the software development communities causing social debt [5], [6]. Social debt, much like technical debt [7] in terms of the impact on software maintenance, refers to unforeseen project costs connected to the presence of non-cohesive development communities having communication or collaboration issues. Such smells may not be the direct cause of software defects, however, they were proved to be refactoring preventers [3].

Corresponding Authors: Zhiqing Shao and Guisheng Fan.

Community smells were evaluated and predicted at the granularity of community sub-groups [1], [3], [8]–[11]. Differently, restructuring smelly communities rely on social efforts of every community member [12]. Empirical guidelines of refactoring community smells, including monitoring, mentoring, planning, exercising, were proposed to shepherd the community [12]–[14].

However, open source software development is known for its distributed and egalitarian nature [15]. Meanwhile, developers focus on primarily their working state and current tasks rather than other issues [16]–[19]. Different from the situations in hierarchical and centralized organizations, the orders and arrangements of community shepherds (*e.g.*, core members [20], architects [13]) may not be strictly executed and followed [15]. Moreover, shepherds may be leaving [20], or even missing [21] from communities. Consequently, general guidelines of software quality that lacked developer-oriented adaptive approaches may be ineffective in practice [17].

Since community smells root in developers' motifs [3], research showed personalities could also influence community smells [14]. Sentiment is an expression of personality [22], which can significantly affect the quality of work [23]–[27], and they were proved to have impacts on a variety of tasks including issue reopening [28], commit changes [29], [30], code refactoring [31], and security [32]. Previous research suggested sentiment-aware classifiers should be built to support developers' work [27].

To the best of our knowledge, there lacks a community smell prediction model designed individually for developers. Moreover, existing quantitative research focused on analyzing the impact of community smells on code quality rather than social representations of developers. Thus, such models cannot support developers to prevent smell from occurring. As a result, we intend to improve prediction approaches and refactoring guidelines of community smells in the granularity of developers individually by involving their sentiments.

This paper builds a developer-oriented and sentiment-aware community smell prediction model. We develop machine learners upon a developer sentiment dataset [33] to predict if a developer is affected by 3 common community smells, such as *Lone Wolf*, *Organization Silo* and *Bottleneck* [1]. Furthermore, the model also predicts if a developer is a *Smelly Quitter* [2], *i.e.*, quitted the community after being affected by any

community smell concerned. Finally, discussions are made on the difference between smelly and non-smelly developers. We analyze the significance of the difference in sentimental features' distributions as well as its effect size.

The main contributions of the papers are listed as follows:

(1) To our knowledge, the first machine-learning based approach to integrate developer sentiments into community smell prediction, as well as the first work to predict community smells' occurrence on developers individually. The proposed model achieved a cross-project prediction performance of F-Measure ranging from 84% to 93%, and a within-project performance of F-Measure ranging from 76% to 91%;

(2) We investigate features having the most predictive power. We discover that 6 sentimental features, *i.e.*, imperative and indicative expressions, politeness, and several emotions, are stronger predictors than the activeness metrics.

(3) We evaluate statistically the relationship between community smell and sentiments, which lead us to the conclusion that developers should communicate in a straightforward and polite way to ensure community healthiness;

(4) We provide an online appendix [34] with the generated dataset to replicate and extend our work.

The rest of this paper is organized as follows: In Section II we summarize related literature. Section III presents how we construct our dataset, while Section IV outlines the settings and research questions, as well as the concerned evaluation metrics. In Section V we discuss the results of this study, while Section VI overviews the threats to the validity of the study and our effort to cope with them. Finally, Section VII concludes the paper and describes future research.

II. Related Work

This section describes researches related to two aspects of this paper, *i.e.*, community smell and developer sentiment.

A. Community Smell

Tamburri, Palomba *et al.* contributed a series of researches [1], [3], [8], [9], [12], [14], [35], [36] concerning the definition [1], [9], detection [1], diffuseness [1], [35], and variability [14] of community smells, as well as their impact on software maintainability [3]. The authors treat community smells as patterns of motifs over collaboration and communication graphs, and they implemented a detection tool called CODEFACE4SMELLS by extending a socio-technical analysis tool called CODEFACE. The tool accepted the input of development mailing list and software repository history information to detect 5 community smells, namely *Organisational Silo*, *Black Cloud*, *Lone Wolf*, *Bottleneck* and *Prima Donnas*. The authors also validated qualitatively [1] the acceptance of the detection result, and they discovered the results were all true positives [8]. Furthermore, the authors observed that community smells could be the preventers of refactoring, and they also intensify code smells continuously [3]. Alternatively, the authors also discovered the associations between community patterns [36], *e.g.*, *Formal Group*, and *Informal Network*, to specific community smells.

In terms of analyzing community smell in the granularity of developers individually, Palomba *et al.* also provided practitioners with refactoring suggestions and frameworks [12]. In a recent study [14], the authors pointed out that communicability is important to prevent community smells, and developers' personality plays an important role in producing smells. Alternatively, Ahammed *et al.* [37] made an exploratory study on 7 Apache projects and discovered the commit activities of developers correlate with their involvement of *Missing Links* (*Lone Wolf*) community smell.

The prediction of community smells has also been actively studied by the research community. Palomba and Tamburri [8] built a state-of-the-art model to predict community smells' emergence on within- and cross-project scenarios. The research also pointed out socio-technical congruence, communicability and turnover-related metrics are the most powerful predictors. Almarimi *et al.* [10], [11] combined Ensemble Classifier Chain (ECC) and Genetic Programming (GP) techniques to predict the emergence of community smells, and they also introduced 4 novel community smells, *i.e.*, *Solution Defiance*, *Sharing Villainy*, *Organizational Skirmish*, and *Truck Factor Smell*.

The major differences of this work to the above-mentioned smell prediction papers are: (1) we predict the occurrence of smells in the granularity of developers individually rather than sub-groups or communities; (2) we involve developers' sentiment features to build predictors, and we assess their predictive power and statistical characteristics in software projects, which were not covered in previous researches.

B. Developer Sentiment

Ortu *et al.* [23]–[26], [33] constructed a multi-aspect developer sentiment dataset based on JIRA comments and sentences, which is regarded as one of the golden dataset [38] for sentiment analysis of the software engineering domain. The authors also analyzed the impact of sentiments. For example, they found impolite comments [25], [39] and bullies [24] are related to longer issue fixing time. Meanwhile, certain combinations of VAD [26] (Valence-Arousal-Dominance) scores may indicate longer issue resolution time, as well as productivity problems such as burnout. Such results in line with Khan *et al.*'s research [40] reporting developers performs better with higher degree of arousal and valence after doing physical exercises. In further research on GITHUB comments [41], the authors also managed to differentiate comments made by users and developers using their emotions, sentiments and degree of politeness.

Happiness is, in most cases, related to positive outcomes of software engineering [42], [43]. However, too many positive comments may also lead to buggy code [30]. Graziotin *et al.* [43], [44] identified 49 consequences of unhappiness in their qualitative research. Internal consequences include low motivation, low cognitive performance, and work withdrawal. External ones contain low code quality and code discharging, *i.e.*, destroying the codebase. Studies also showed that negative emotions are signals of the appearance of bug fix-inducing

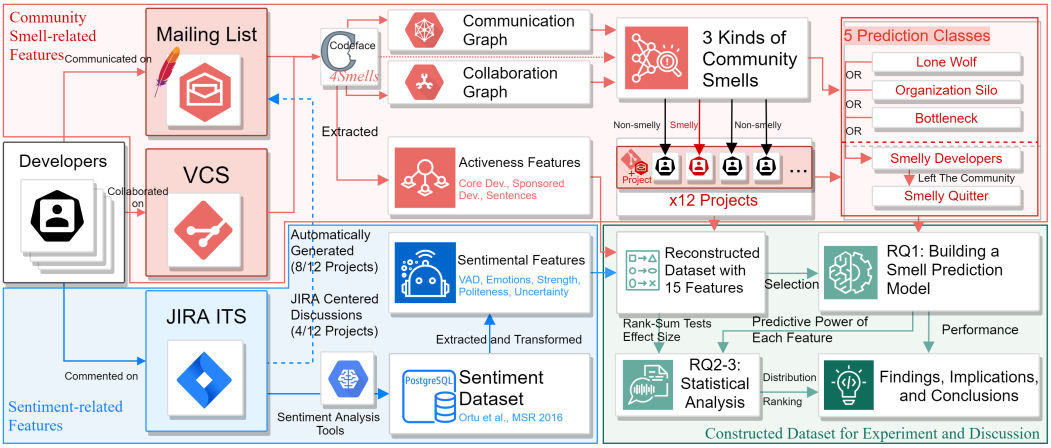

Fig. 1. Dataset construction and its usage.

changes, but they may result in a longer issue-fixing time [30] and issue-reopening [28]. As for negative sentimental expressions in code refactoring, Singh *et al.* [31] concluded that poor tooling, error-proneness and high complexity of such activities could cause negative sentiments. Controversially, Islam *et al.* [45] reported opposite observations. Additionally, they discovered that the sentiment of code commit varies from commit types, *e.g.*, there exist higher negative emotions for new feature implementation.

To sum up, it is possible to detect developers' sentiments, and sentiments do have affections on the software development process. However, a generally-accepted theorem explaining the pattern of such affection has not been proposed at present.

III. Dataset Construction

This section describes how we export and process the developers' sentiment features and community smells' occurrences to define and train our machine learner in the next section. Fig. 1 depicts generally the origins and construction methods of our dataset, as well as its further usage in this paper.

A. Extracting Developer Sentiment Features

First, we recover and adjust the raw data, *i.e.*, sentiments of every developer of specific projects in JIRA ITS located in 2 tables called `jira_issue_comment` and `jira_user`. Afterwards, we also group the features by developers and projects to prepare for cross- and within-project prediction. Details of the features will be described in Section IV.

B. Selecting Projects and Fetching Mailing Lists

The sentiment dataset consists of project comments in ITS of 4 major open source foundations namely ASF, Codehaus, JBoss, and Spring. However, only 23 projects contain sentiment data. Among the 23 projects, we first exclude 3 projects whose mailing list service providers do not support archive extraction. Next, we exclude 4 more projects as their mailing lists do not cover the time range of their sentiment data. Afterwards, we remove 3 software projects as they were sharing the same instances of JIRA ITS or Version Control System (VCS) repository with other projects, and such projects were incompatible with Codeface4Smells. Finally, We also filter out 1 project whose VAD data is missing in the dataset.

To sum up, the actual 12 projects we use to perform the analysis are listed in Table I. We fetch their mailing lists from the archives provided by their open source foundations.

C. Detecting community smells

Although we are aware of the existence of other community smell detection tools or models [10], [11], they are not publicly available. Thus, we apply the state-of-the-art tool Codeface4Smells [1], [9] to detect community smells.

We follow strictly the instructions of the Codeface4Smells repository, *e.g.*, we execute the application in the suggested Vagrant instance, and we fix the broken dependencies in order to avoid platform-specific problems. The only modification is adding exportation features to the socio-technical analysis script to derive names and e-mails of developers affected by each community smell.

As for configurations, community smell analysis must be performed in a given window. In our case, the window is 3 months as previous works suggested [1], [8], [9]. Similar as the tool's replication package did [9], we also specify every commit to analyze in configuration files, the commits are exported from Git repositories of the projects using PyDriller [46], which is also a commonly applied tool in software repository mining tasks [17]. The configuration files are also provided in our replication package [34].

Codeface4Smells is able to detect 5 community smells, including *Organization Silo*, *Lone Wolf*, *Bottleneck*, *Black Cloud*, and *Prima Donnas*. However, *Prima Donnas* detection is not empirically [1] proved effective since its first appearance [9], so we do not take this community smell into consideration. *Black Cloud* is sparsely distributed in software systems [1]. In our dataset, there are several *Black Cloud* appearances. However, the developer affected is not captured in the sentiment dataset, thus we could not perform *Black Cloud* prediction. Consequently, our research scope includes

TABLE I
THE PROJECTS ANALYZED IN THIS PAPER

Repository	Project Name	Developers	Date Range	Issue Reports	Mailing List Name
ASF	HBase	919	2007-04 - 2014-04	68806	hbase-dev[1]
ASF	Hadoop Common	1221	2009-05 - 2014-02	61905	commons-dev[1]
ASF	Hadoop HDFS	745	2009-05 - 2011-04	42188	hadoop-hdfs-dev[1]
ASF	Cassandra	1161	2009-03 - 2014-03	41937	cassandra-dev[1]
ASF	Hadoop Map/Reduce	857	2009-05 - 2011-03	34747	hadoop-mapreduce-dev[1]
ASF	Hive	839	2008-09 - 2014-03	34449	hive-dev[1]
ASF	Harmony	306	2005-09 - 2011-07	28325	harmony-dev[1]
ASF	OFBiz	538	2006-07 - 2014-02	25667	ofbiz-dev[1]
JBOSS	Hibernate ORM	3958	2007-06 - 2014-01	23549	hibernate-dev[2]
ASF	Camel	882	2007-04 - 2014-01	21758	camel-dev[1]
ASF	Wicket	1210	2006-10 - 2014-01	17030	wicket-dev[1]
ASF	Zookeeper	484	2005-09 - 2011-07	13634	zookeeper-dev[1]

3 community smells, namely *Organizational Silo*, *Lone Wolf*, and *Bottleneck*. We also involve *Smelly Developers* and *Smelly Quitters* as additional prediction classes, the details will be described in section IV.

IV. DESIGN AND EVALUATION OF A DEVELOPER-ORIENTED AND SENTIMENT-AWARE COMMUNITY SMELL PREDICTION MODEL

The goal of our study is to evaluate to what extent the occurrence of community smells on developers can be predicted by their sentiments, with the purpose of understanding the impact of sentiment to community smells from the granularity and perspective of developer themselves, in the mean time, provide researchers and practitioners with novel aspects and suggestions on community healthiness. This section proposes our research questions and our methodology.

A. Research Questions and Methodologies

> RQ1: To what extent can we predict the occurrence of community smells on developers using their sentiments?

Starting from the dataset we build in Section III, we define dependent and independent variables, and we build prediction models using Machine Learning classifiers. To pick the most appropriate machine learner, and to figure out why our model works or fails, their performances must be assessed. Hence, we propose RQ2.

> RQ2: What is the predictive power of sentiments to the occurrence of community smells on developers?

We investigate further the predictive power of each feature to reveal the contribution of every sentimental feature to our prediction model. The conclusion may provide us with useful information to explain our model and to look deeply into the distribution of sentimental features, which leads us to RQ3.

> RQ3: Are smelly and non-smelly developers different in terms of their sentiments?

Apart from model performance metrics, we intend to explain how sentiments impact the healthiness of development community according to their statistical characteristics including data distributions and mean values, and to differentiate smelly and non-smelly developers. Furthermore, we attempt to make suggestions to practitioners based on our findings.

B. RQ1: Defining and Evaluating the Proposed Model in Cross- and Within-project Scenarios

1) Dependent Variables: In this paragraph, we list the definition of 5 smell-related prediction classes of our model. The 3 concerned community smells are defined as follows:

- **Organizational Silo**: refers to the presence of siloed areas of the developer community that do not communicate, except through one or two of their respective members [1], *i.e.*, co-committing developers do not directly communicate at all [9].
- **Lone Wolf**: reflects co-committing software developers who exhibit uncooperative behaviour and mistrust by not appropriately communicating [1], *i.e.*, the collaboration edges that do not have a communication counterpart [9].
- **Bottleneck**: unique boundary spanners interpose themselves into every interaction across sub-communities [1].

Despite the 3 smells, we also export 2 related classes:

- **Smelly Developer**: developers affected by any of the 3 above-mentioned community smells.
- **Smelly Quitter**: *Smelly Developers* from the last analysis window who quit the community in current window [8].

Fig. 2 illustrates an example of community smell detection, in which smelly developers are marked in red. The left part of Fig. 2 shows a siloed area of developers, while the right part illustrates 2 lone wolves collaborating on shared code with indirect communication. To clarify, *Organizational Silo* is a subset of *Lone Wolf* [9]. Technically, the major difference between the two smells stands in the definition of lacking connectivity in the communication graph. *Organizational Silo* is detected if collaborating developers are disconnected in the

[1]http://mail-archives.apache.org/mod_mbox/{Mailing List Name}
[2]https://lists.jboss.org/archives/list/hibernate-dev@lists.jboss.org/

TABLE II
FEATURES EXTRACTED FROM THE DEVELOPER SENTIMENT DATASET AND CODEFACE4SMELLS-DEFINED METRICS

VAD. Automatically detected, the summation of mean value of sentences in comments by extending lexicons [26] in SENTISTRENGTH [47].		
Mean Valence	VAL	The mean intensity of valence, *i.e.*, how developers enjoy a situation.
Mean Arousal	ARO	The mean intensity of arousal, *i.e.*, increased alertness.
Mean Dominance	DOM	The mean intensity of dominance, the extent that developers were feeling in control.
Emotions. Binary value labeled manually in the dataset.		
Mean Sadness	SAD	The mean intensity of all sadness expressions.
Mean Anger	ANG	The mean intensity of all angry expressions.
Mean Love	LOV	The mean intensity of all love expressions.
Mean Joy	JOY	The mean intensity of all joy expressions.
Sentiment Strength. Measured using SENTISTRENGTH [47], ranges between [-1;1].		
Mean Postive Sentiment	POS	The mean intensity of all sentiments smaller than 0.
Mean Negative Sentiment	NEG	The mean intensity of all sentiments greater than 0.
Politeness. Binary value measured using Danescu *et al.*'s tool [48].		
Politeness Proportion	POL	The proportion of polite expressions in all commentary sentences of a developer.
Uncertainty. 4 categorical Grammatical Moods (GM) and modality in [-1;1] measured using PATTERN [49] by detecting auxiliary verbs and adverbs.		
Indicative GM Proportion	IND	The proportion of sentences that express fact or belief, *e.g.*, It rains.
Imperative GM Proportion	IMP	The proportion of sentences that express command, warning, *e.g.*, Do not rain!
Conditional GM Proportion	CON	The proportion of sentences in the form like would, may, or will, *e.g.*, It might rain.
Subjunctive GM Proportion	SUB	The proportion of sentences in the form like wish, were, *e.g.*, I hope it rains.
Mean Degree of Modality	MOD	The degree of uncertainty of a sentence, ranges between [-1;1].
Activeness. Control variables reflecting developers' working state other than sentiments.		
Total Sentences	SEN	Number of total senteces developers commented in the JIRA ITS, extracted from [33].
Core Ranges Count	COR	Number of total ranges developer acted as a core developer, measured by CODEFACE4SMELLS [1], [9].
Sponsored Ranges Count	SPO	Number of total ranges developer commited only in working hours [8], measured by CODEFACE4SMELLS.

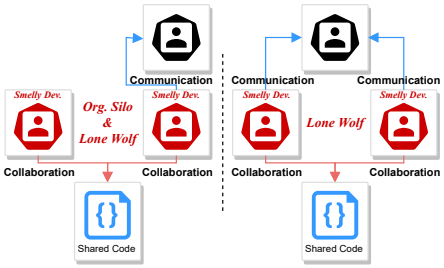

Fig. 2. Lone Wolf and Organizational Silo identification pattern.

communication graph. *Lone Wolf* is detected if collaborating developers are not neighbours in the communication graph, *i.e.*, lacks 1-degree connection. Differently, *Bottleneck* is detected based on completely the communication graph, and its definition is remarkably similar to the unique boundary spanner in social-network analysis [1].

Our motivation to involve *Smelly Quitter* prediction is that literature reported negative sentiments may cause developer to "destroy codebase" [43], [44], and to "quit over mistreatment" [50]. Similarly, we assume it may also lead to the discharge of community.

2) Independent Variables: We integrate sentimental and activeness features in Table II as independent variables.

Similar to [45], we split positive and negative sentiment into 2 features, POS and NEG.

We do not include an impoliteness proportion feature because the politeness labels available in the dataset are either polite or impolite, which will result in high correlation if we involve both. We ignore the confidential of politeness score since Ortu *et al.* [33] have already dropped the results with confidential less than a conventional threshold (0.5) [25].

The dataset also contains a mood column whose proper meaning was not clearly described in the original paper [24]. We look into the documentation of the detection tool [49] and confirm this feature measures the GM of a sentence, and the result is mapped into 4 classes, *i.e.*, {indicative, imperative, conditional, subjunctive}. In the dataset the 4 classes are presented in 4 values, namely {0,1,2,3}. We map the mood attribute into 4 proportional features to measure the developers' characteristics of expression.

Finally, we also introduce 3 activeness features including total sentences of developers commented on ITS (SEN) [33], the count of ranges that developers acted as core (COR) and sponsored (SPO) developers [1], [8]. They are used as control variables to assess the actual impact of sentimental features in comparison with activeness.

3) Data Balancing and Feature Selection: Our dataset is highly imbalanced, which in line with the observations in related researches [8], [10]. Smelly developers account for 4.80% of the overall developer population, which may hinder model performance. Therefore, we preprocess our data with Random Undersampling strategy, which is proved beneficial to imbalanced data in software engineering [51]. We also address potential multicollinearity problem by removing the correlated features as previous researches proposed [8].

4) Scenarios: We build models separately in both cross-project and within-project scenarios. For cross-project tasks, we merge the developers' features regardless of the projects they are working on. Notably, the same developers appearing on multiple projects are treated differently.

5) Training Machine Learners: Previous researches in community smell and code smell detection showed Random Forest was the best-performed classifier [8], [17]. The reason we evaluate again the different classifiers is that our data derive

from developers individually, which is different from previous scenarios. Moreover, predictors have different characteristics in terms of design and effectiveness, *i.e.*, the risk of overfitting and different execution speed [8]. We intend to discover the best classifier for our scenario.

We apply the SCIKIT-LEARN package [52] from PYTHON to train machine learners using multiple classifiers that have been used in previous works [8], [17], [53], including Random Forest, Decision Tree, Support Vector Machine, Multilayer Perceptron, Adaboost, Naive-Bayes, and Logistic Regression.

As related work suggested [17], instead of using default settings, we configure the hyper-parameters of the classifiers by exploiting Exhaustive Grid Search with a 10-Fold Cross-Validation strategy to calculate multiple times the performance of every combination of parameters.

6) Validation and Assessment: To validate the performance of each classifier, we employ a 10 × 10-Fold Cross-Validation strategy to make sure that we have unbiased and stable [8], [54] performance data in cross-project scenarios. For the within-project scenario, we apply the Leave-One-Out Cross-Validation (LOOCV) according to [8], [54] which is proved to be stable and the least biased one in this scenario. Finally, we compute classical performance metrics including precision, recall, F-Measure, and AUC-ROC [55] to pick the best-ranked classifier in both scenarios. Thus, we could answer RQ1 quantificationally.

C. RQ2: Explaining the Predictive Power of Features

To answer this RQ, we need to explain the extent of predictive power that each independent variable contribute to the best-performed classifier in RQ1.

First, we apply an information gain algorithm, which has already been used in previous researches of community smells [8] and code smells [17] to compute the gain provided by each feature within the model to the correct prediction of a dependent variable. The approach measures the amount of reduction in uncertainty, *i.e.*, Shannon's Entropy, of the model concerned after involving in a new feature. This paper applies the mutual information classification [56] implementation of SCIKIT-LEARN package, which is originally designed for feature selection and identical in definition to the Gain Ratio Feature Evaluation used by previous researches [8], [17]. The algorithm ranks the features in descendent order according to each of their contribution to the entropy reduction of the concerned model. Thus, the features contributing the most predictive power could be identified.

Then, in order to assess the significance of the ranking of information gain, we also involve the Scott-Knott Effect Size Difference (SK-ESD) test. Scott-Knott test [57] is a statistical measure to compare and differentiate model performance using a hierarchical clustering approach to group the means of assessment metrics, *e.g.*, F-Measures of multiple models. Scott-Knott test assumes input data distribution to be normally distributed, thus may be ineffective for non-normally distributed data. The ESD test is an enhanced version of the original Scott-Knott test that corrects the non-normal distribution of the input

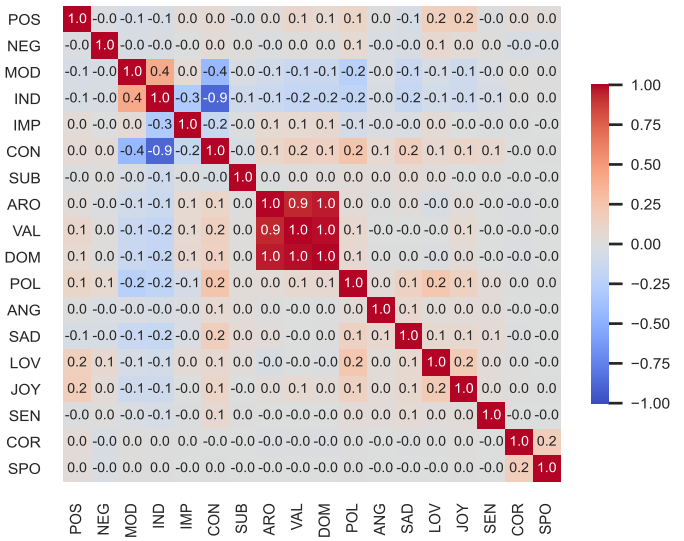

Fig. 3. Correlation heat-map of features.

to make it comply with the requirements executing the Scott-Knott test. Meanwhile, it uses Cliff's Delta [58] as an effect size measure to merge groups having negligible effect sizes. Scott-Knott and SK-ESD tests have already been applied in various researches in code smell [17], community smell [8], and software defect [59]. We use the original implementation of Tantithamthavorn *et al.* [54], [60] available on GITHUB.

Both information gain and SK-ESD algorithms are executed on each prediction class independently. We report the results from both cross- and within-project scenarios.

D. RQ3: Investigating the Difference between Smelly and Non-Smelly Developers in Terms of Sentiments

We take a closer look at the sentimental features' difference in distribution and intensity for smelly and non-smelly developers. We apply statistical measures, *i.e.*, Wilcoxon Ranksum Test ($\alpha = 0.05$, $p < \alpha$ for statically significant [61]) to analyze the significance of the difference in distribution. Meanwhile, we calculate Cliff's Delta (δ) for each test to measure the effect size, *i.e.*, the extent of the difference, which is negligible when $|\delta| < 0.147$, small when $0.147 \leq |\delta| < 0.33$, medium when $0.33 \leq |\delta| < 0.474$, and large otherwise. Such measures have been applied in previous works of developer sentiment [26], [41] and code smell [61]. We also report the mean and variance values of the features.

We expect to find statistical significance from our dataset to support our findings in RQ2, and to provide practitioners with advice to prevent community smells from occurring at an early stage.

V. RESULT AND DISCUSSION

In this section, we demonstrate and discuss the results of our experiment, answer the proposed research questions, and describe our findings.

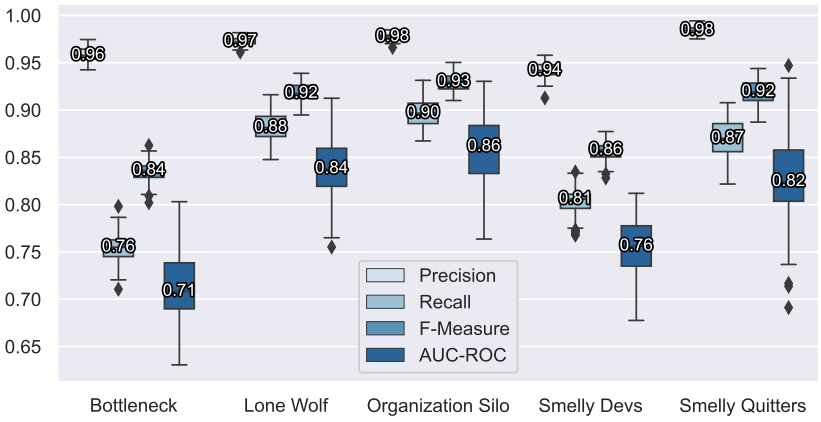

Fig. 4. Cross-project prediction performance.

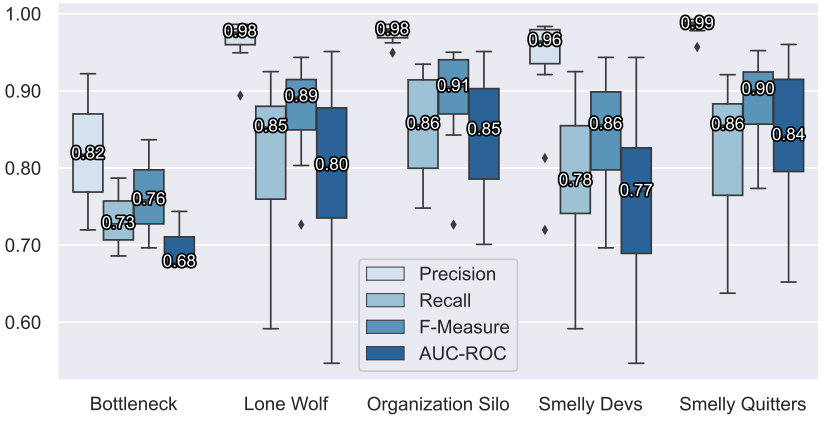

Fig. 5. Within-project prediction performance.

A. RQ1: Model Performance

In Fig. 3, we assess the correlations of our features. Result shows the VAD features, *i.e.*, VAL, ARO, DOM, correlate with each other (ρ>0.9). Although we find removing any 2 of them does not cause a significant change in the classifier's performance (<2%), we still exclude VAL and DOM to avoid potential multicollinearity problems. We also exclude CON for the same reason. Consequently, we remove 3 features out of 18 original ones.

We train multiple classifiers on the dataset and report the result of the best performed one, *i.e.*, Random Forest, in Fig. 4 and Fig. 5 for both cross- and within-project prediction, and we include results for the 2 scenarios in the sets of box-plots. We also present the median of weighted average performance of our models in Table III.

Due to inconsistent community smell appearance in different projects, the performance of within-project prediction is less stable than cross-project ones, revealing potential drawbacks of our approach such as the capability to deal with cold start problem, *i.e.,* do not have sufficient useful data to start with, which should be addressed in future work.

However, the median values of the model's performances show that the model still works in most cases.

> Finding 1. Our prediction model could predict the occurrence of community smells on developers in most cases. In terms of cross-project prediction, our model reaches mean F-Measures ranging from 84% to 93%. Meanwhile, the model achieves F-Measures from 76% to 91% for within-project prediction.

B. RQ2: Features' Predictive Power

Table IV lists the information gains that features contribute to our prediction model in descendent order. Metrics in bold and italic are ***Sentimental Features***. Column W.-P. & C.-P.

list the SK-ESD rank of gain of features to the within- and cross-project model. Fig. 6 and Fig. 7 depict the mean (in solid lines), median (in dashed lines), as well as the rank of each feature's gain. Metrics given the same rank are marked in the same color. The ranks of information gains in different classes are similar, so we do not report them due to the limitation of space. Therefore, the gains presented in this section is to describe the contribution of metrics to the overall performance of our model.

Every sentimental feature has some contribution to the model (mean Gain Ratio ≥ 0.033), and their contributions are average and moderate. Meanwhile, the rankings are slightly different in the 2 scenarios. However, 6 metrics, *i.e.*, IMP, SAD, POL, IND, LOV, JOY, are always stronger predictors than any of the activeness features. Meanwhile, they are also among the highest ones.

In particular, imperative GM is the strongest factor among all the features in terms of predictive power, while indicative GM is in the same rank, showing certainty is a determining aspect of community healthiness.

Either positive or negative emotions, including sadness, love, and joy, are related to community smell issues.

POL came after IMP and SAD, which is in line with researches [24], [25], [39], [48] suggesting politeness may have an impact on developers' productivity.

The above-mentioned sentimental features contribute more than the activeness ones, *i.e.*, SEN, COR and SPO, indicating the centrality and discussion activeness of developers are weaker predictors than several sentiments. Recent work [37] also reported that the number of commits are related to community smells' occurrence. In response, we are planning to investigate the traditional process metrics' impact on developers' working state in the software community. Since COR measures the centrality of developers in the communication and the collaboration graphs, we assume it is likely to be correlated with commit activeness over software systems.

> Finding 2. In terms of community smells' occurrence on developers, 6 sentimental features are stronger predictors than activeness ones. Imperative and indicative GM are among the strongest predictors, indicating certainty a key factor to community healthiness. Politeness and both positive and negative emotions including sadness, love, and joy, are also powerful predictors.

C. RQ3: Smelly vs. Non-Smelly Developers

Fig. 8 depicts the distribution of sentimental features using an enhanced version of box-plot called boxen-plot [62], which is more capable of displaying tails of large-sampled data, as it cuts data into more quantiles.

Table V lists the result of statistical analysis in order to differentiate the sentimental characteristics derived from our dataset, in which {Y,N} represent smelly and non-smelly Developers. Effect sizes in {Large, Medium, Small, Negligible} are Mapped to {L, M, S, -}. Since we analyze the distribution of metrics' data separately, zeros are removed for each group of metrics in this section to ensure the effectiveness of data.

Since predictive power could speak for the actual impact of features on the prediction outcome [8], [17], we reveal IMP, IND, POL, SAD, LOV, and JOY may have impacts on community smell, which would be analyzed and discussed in the next paragraphs.

The IMP feature is the top-ranked feature in terms of predictive power. From experience, we assume instructive expressions are more likely to be serious and impolite, which

TABLE III
WEIGHTED AVERAGE OF MODEL PERFORMANCE

Class	Scenario	Prec.	Rec.	F-Meas.	AUC-ROC
Bottleneck	Within	.96	.76	.84	.71
	Cross	.82	.73	.76	.68
Lone Wolf	Within	.97	.88	.92	.84
	Cross	.98	.85	.89	.80
Org. Silo	Within	.98	.90	.93	.86
	Cross	.98	.86	.91	.85
Smelly Dev.	Within	.94	.81	.86	.76
	Cross	.96	.78	.86	.77
Smelly Quitter	Within	.98	.87	.92	.82
	Cross	.99	.86	.90	.84

TABLE IV
RANKING AND MEAN GAIN RATIO OF EACH FEATURE.

Metric	Mean	W.-P.	C.-P.	Metric	Mean	W.-P.	C.-P.
IMP	.18	1	1	***ANG***	.12	4	4
SAD	.18	1	1	***POS***	.11	5	5
POL	.18	1	1	***MOD***	.08	6	6
IND	.18	1	1	***SUB***	.07	7	7
LOV	.18	1	2	***ARO***	.03	8	8
JOY	.17	2	1	COR	.03	9	9
NEG	.13	3	2	SPO	.00	10	10
SEN	.13	3	3				

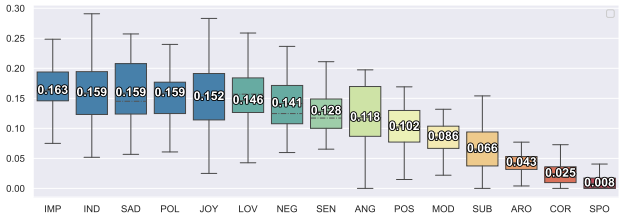

Fig. 6. Within-project Information gain classified by SK-ESD.

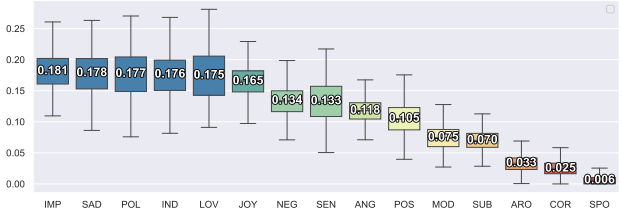

Fig. 7. Cross-project Information gain classified by SK-ESD.

TABLE V
RESULT OF STATISTICAL ANALYSIS.

Features	**IMP**		**POL**		**LOV**		**JOY**		**NEG**		**IND**	
Wilcoxon R.	p<0.001		p<0.001		p<0.001		p<0.001		p<0.001		p<0.001	
Cliff's Delta	-0.48	L	-0.25	S	0.34	M	0.38	M	-0.09	-	-0.27	S
Smelly Dev.?	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**
Mean	0.09	0.26	0.54	0.67	0.14	0.13	0.08	0.02	-0.16	-0.17	0.65	0.77
Variance	0.01	0.08	0.04	0.08	0.04	0.09	0.06	0.03	0.01	0.02	0.33	0.06
Features	**SAD**		**ANG**		**POS**		**MOD**		**SUB**		**ARO**	
Wilcoxon R.	p<0.001		p<0.001		p<0.001		p>0.05		p<0.001		p<0.001	
Cliff's Delta	0.13	-	-0.50	L	0.15	S	-	-	-0.53	L	-0.08	S
Smelly Dev.?	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**
Mean	0.27	0.33	0.03	0.19	0.21	0.20	0.58	0.57	0.03	0.13	1.00	1.03
Variance	0.08	0.24	0.00	0.13	0.01	0.02	0.02	0.06	0.01	0.06	0.03	0.10

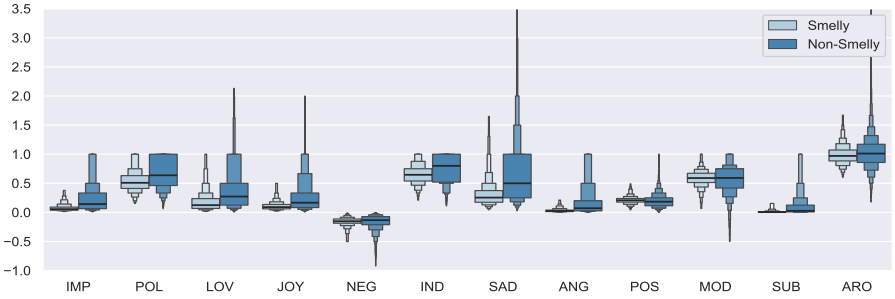

Fig. 8. Distribution of the features.

would become an obstacle for cooperation. However, result shows the distribution of the proportion of imperative GM is significantly different for smelly and non-smelly developers with a large effect size. Result also reveals non-smelly developers use 3 times more frequently the imperative expressions, *i.e.*, commands and warnings. Moreover, indicative GM is also among the top contributing features, and smelly developers use less indicative expressions than non-smelly ones. Hence, we conclude that ensuring certainty is vital to developers' communication and collaboration quality. This conclusion is in line with the observations made from purely technical aspects [63]. Interestingly, non-smelly developers also use more subjunctive expressions to express wishes and opinions, indicating the importance of certainty may differ from scenarios.

Difference in politeness, however, does not have a great effect size as we expect, *i.e.*, in our dataset, smelly developers communicate in a similar structure compared with non-smelly developers in terms of politeness. Nevertheless, non-smelly developers use 24% more frequently the polite expressions in general than smelly developers.

Previous researches [42], [64] in developers' sentiments topics mainly regard happiness as a positive factor to development task. Surprisingly, expressing more joyful emotion may also indicate community smells' occurrence. Meanwhile, we also notice literature reporting that too many positive comments [30] indicate potential bugs. In contrast, negative emotions have always been an indicator of potential problems in software engineering [27]. Unexpectedly, except for anger, no notable difference are found in smelly and non-smelly developers in terms of negative sentiments. According to their mean values, non-smelly developers are even more bad-tempered and depressed than the smelly ones. Due to the multi-facet and complex nature of sentiments and development activity, the difficulties in interpretation occurred frequently in developer sentiment analysis [20], [25], [26], [28], [65].

Practically, we suggest developers should communicate in a straightforward and polite way. As for emotions, we cannot provide suggestions until further empirical investigation and case studies are made to figure out positive and negative emotions' actual impact. Indeed, the unforeseen part of results shed light on the necessity of a deeper understanding of developers' sentiments and their impact on software community through qualitative and quantitative study. Beyond technical aspects, the research community need to improve and reshape the framework of comprehending developers' perception [17]–[19], [66] as well as their task context [16], [67]–[69].

Finding 3. Smelly developers are different from non-smelly ones in terms of sentiments. They are less polite, and they use less imperative and indicative GM, *i.e.*, less certain statements, instructions and warnings. Unexpectedly, they express more positive and less negative emotions. To ensure community healthiness, we suggest developers should communicate in a straightforward and polite way.

VI. Threats to Validity

This section clarifies the way we address threats to validity.

A. Construct Validity

The major threat to construct validity is the reliability of our datasets. We combine 2 sources of information, *i.e.*, community smell detection results and developer sentiment dataset.

In terms of community smell detection, we employ an open-source tool called CODEFACE4SMELLS [1]. The authors provided detailed replication data to prove the dependability of the tool, *i.e.*, its output were all true positives. Hence, we believe the tool is reliable. In addition, we follow strictly the installation, configuration, and execution guides [9] of the detection tool. As for software repositories and mailing lists, we fetch them from original sources of ASF and JBOSS. To a great extent, we can confirm the reliability of this source.

The developer sentiment dataset is proposed and improved progressively by Ortu *et al.* [23]–[26], [33] through years of validated works. Except for the emotions (joy, love, anger, sadness), all data are automatically detected by lexicon-based tools. The performance of the tools may be a threat to validity. Indeed, no tool is ready for detecting sentiments in all kinds of discussions in software engineering [70], *e.g.*, app reviews and Q&As. However, tools employed by the dataset are all state-of-the-arts, *e.g.*, SENTISTRENGTH [47] achieved an agreeable performance ranging from 0.70 to 0.99 in terms of positive and negative sentiment detection [71] in the JIRA ITS, and its reliability have also been proved in other researches [38], [72]. Thus, we conclude that the reliability of the sentiment dataset in the context of our research is acceptable.

The process of combining the two datasets, however, may cause some loss in information. To match the developers in both datasets, we make our best effort to link developers from both sides with their e-mails and names. However, 8.8% of the smelly developers are not found in the sentiment dataset. Thus, their data are dropped. Nevertheless, we still manage to preserve most of the data.

The coherence of mailing lists and developer comments in JIRA may be a threat as well. Therefore, we investigate the contents of mailing lists. In most of the cases (8 out of 12), they are automatically generated JIRA discussions. If not so, they are JIRA-centered discussions, *i.e.*, comments attaching hyperlinks of JIRA tickets. The involvement of ITS contents in mailing lists was also observed in other research [73]. In conclusion, we are measuring discussions in the same context presented in different layouts.

B. Conclusion Validity

The mix of manually labeled emotions and automatically detected sentiments in the dataset may hinder the practical value of our model, as the evaluation of sentiments is not fully automatic. However, such labels could be replaced by the outputs of reliable sentiment evaluation tools [74], [75].

In terms of the reliability of model settings, we configure the hyper-parameters using Grid Search, and we employ LOOCV as well as 10×10-Fold Validation, which was proved stable by previous research [54]. We also report results of classical evaluation metrics, *i.e.*, precision, recall, F-Measure, and AUC-ROC. Furthermore, we apply statistical tests, *e.g.*, SK-ESD and Effect Sizes, to validate the significance of our conclusions.

C. External Validity

Since different projects may involve developers in various backgrounds [35], [76], the multi-faceted nature of software development and developer sentiment is an unavoidable threat to external validity. To address this issue, we perform our study in 12 active open-source projects of 2 major open source ecosystems to maximize the generality of our conclusion. Such systems have been widely studied in previous works of software engineering [17], [53], [77].

VII. Conclusion

This paper investigates whether, and to what extent, the occurrence of community smells on developers could be predicted by their sentiments. Furthermore, it analyzes the difference of smelly and non-smelly developers in terms of the distribution and intensities of their sentiments.

We construct sentimental features from a developer sentiment dataset [33] as independent variables. Meanwhile, we exploit CODEFACE4SMELLS [1], [9] to generate dependent variables concerning whether a developer is affected by 3 community smells, *i.e.*, *Lone Wolf*, *Organization Silo*, and *Bottleneck*, and if he/she is a *Smelly Quitter*. Afterwards, we also assess the predictive power of all the features. Finally, we evaluate the significance and effect sizes of the distributions of sentiments on smelly and non-smelly developers.

Result shows our model achieves mean F-Measures ranging from 76% to 93% in within- and cross-project prediction. Additionally, sentimental features including imperative and indicative GM, politeness, sadness, love, and joy are stronger predictors than the activeness metrics. We reveal smelly developers are less polite, and they use less certain statements, instructions, and warnings. Unexpectedly, we also discover that smelly developers express more positive and less negative emotions, which need to be interpreted in further research. To conclude, we suggest developers should communicate in a straightforward and polite way.

Future work includes: (1) the integration of more effort- and process-aware metrics, (2) involving other kinds of discussions such as chat messages [78], (3) interpreting the pattern of positive and negative emotions' interaction with social debt.

Acknowledgment

This work is partially supported by the NSF of China under grants No.61772200, the Shanghai Natural Science Foundation No.17ZR1406900, 17ZR1429700, and the Software and Integrated Circuit Industry Development Special Funds of Shanghai Economic and Information Commission under Grant No.XX-XXFZ-02-20-2463.

REFERENCES

[1] D. A. Tamburri, F. Palomba, and R. Kazman, "Exploring community smells in open-source: An automated approach," *IEEE Transactions on Software Engineering (TSE)*, p. Early Access, 2019. doi: 10.1109/TSE.2019.2901490.

[2] B. Caglayan, B. Turhan, A. Bener, M. Habayeb, A. Miransky, and E. Cialini, "Merits of organizational metrics in defect prediction: An industrial replication," in *IEEE/ACM 37th International Conference on Software Engineering (ICSE)*, vol. 2, pp. 89–98, 2015.

[3] F. Palomba, D. A. Tamburri, F. Arcelli Fontana, R. Oliveto, A. Zaidman, and A. Serebrenik, "Beyond technical aspects: How do community smells influence the intensity of code smells?," *IEEE Transactions on Software Engineering (TSE)*, vol. 47, no. 1, pp. 108–129, 2021.

[4] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, "Refactoring: improving the design of existing code," Addison-Wesley, 1999.

[5] D. A. Tamburri, "Software architecture social debt: Managing the incommunicability factor," *IEEE Transactions on Computational Social Systems (TCSS)*, vol. 6, no. 1, pp. 20–37, 2019.

[6] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "What is social debt in software engineering?," in *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 93–96, 2013.

[7] J. M. Conejero, R. Rodríguez-Echeverría, J. Hernández, P. J. Clemente, C. Ortiz-Caraballo, E. Jurado, and F. Sánchez-Figueroa, "Early evaluation of technical debt impact on maintainability," *Journal of Systems and Software (JSS)*, vol. 142, pp. 92–114, 2018.

[8] F. Palomba and D. A. Tamburri, "Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach," *Journal of Systems and Software (JSS)*, vol. 171, p. 110847, 2021.

[9] D. A. Tamburri, *"An approach to measure community smells"*, Available at: https://github.com/maelstromdat/codeface4smells_TR. Accessed: Jan, 2021.

[10] N. Almarimi, A. Ouni, and M. W. Mkaouer, "Learning to detect community smells in open source software projects," *Knowledge-Based Systems (KBS)*, vol. 204, p. 106201, 2020.

[11] N. Almarimi, A. Ouni, M. Chouchen, I. Saidani, and M. W. Mkaouer, "On the detection of community smells using genetic programming-based ensemble classifier chain," in *15th International Conference on Global Software Engineering (ICGSE)*, p. 43–54, 2020.

[12] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, and F. Ferrucci, "Refactoring community smells in the wild: The practitioner's field manual," in *ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, p. 25–34, 2020.

[13] D. A. Tamburri, R. Kazman, and H. Fahimi, "The architect's role in community shepherding," *IEEE Software*, vol. 33, no. 6, pp. 70–79, 2016.

[14] G. Catolino, F. Palomba, D. A. Tamburri, and A. Serebrenik, "Understanding community smells variability: A statistical approach," in *ACM/IEEE 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, p. unpublished, 2021, Available at: https://fpalomba.github.io/pdf/Conferencs/C58.pdf.

[15] N. Ducheneaut, "Socialization in an open source software community: A socio-technical analysis," *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 4, pp. 323–368, 2005.

[16] N. Sae-Lim, S. Hayashi, and M. Saeki, "Context-based code smells prioritization for prefactoring," in *IEEE 24th International Conference on Program Comprehension (ICPC)*, pp. 1–10, 2016.

[17] F. Pecorelli, F. Palomba, F. Khomh, and A. De Lucia, "Developer-driven code smell prioritization," in *IEEE/ACM 17th International Conference on Mining Software Repositories (MSR)*, p. 220–231, 2020.

[18] N. Sae-Lim, S. Hayashi, and M. Saeki, "An investigative study on how developers filter and prioritize code smells," *IEICE Transactions on Information and Systems*, vol. 101-D, no. 7, pp. 1733–1742, 2018.

[19] N. Sae-Lim, S. Hayashi, and M. Saeki, "How do developers select and prioritize code smells? A preliminary study," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 484–488, 2017.

[20] I. Ferreira, K. Stewart, D. German, and B. Adams, "A longitudinal study on the maintainers' sentiment of a large scale open source ecosystem," in *IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pp. 17–22, 2019.

[21] J. Coelho, M. T. Valente, L. L. Silva, and E. Shihab, "Identifying unmaintained projects in github," in *12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2018.

[22] J. Lin, W. Mao, and D. D. Zeng, "Personality-based refinement for sentiment classification in microblog," *Knowledge-Based Systems (KBS)*, vol. 132, pp. 204–214, 2017.

[23] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," in *IEEE/ACM 11th Working Conference on Mining Software Repositories (MSR)*, p. 262–271, 2014.

[24] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive? empirical study of affectiveness vs. issue fixing time," in *IEEE/ACM 12th Working Conference on Mining Software Repositories (MSR)*, pp. 303–313, 2015.

[25] M. Ortu, G. Destefanis, M. Kassab, S. Counsell, M. Marchesi, and R. Tonelli, "Would you mind fixing this issue?," in *16th International Conference on Agile Software Development (XP)*, vol. 212, pp. 129–140, 2015.

[26] M. Mäntylä, B. Adams, G. Destefanis, D. Graziotin, and M. Ortu, "Mining valence, arousal, and dominance: Possibilities for detecting burnout and productivity?," in *IEEE/ACM 13th International Conference on Mining Software Repositories (MSR)*, p. 247–258, 2016.

[27] S. C. Müller and T. Fritz, "Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress," in *IEEE/ACM 37th International Conference on Software Engineering (ICSE)*, vol. 1, pp. 688–699, 2015.

[28] J. Cheruvelil and B. C. da Silva, "Developers' sentiment and issue reopening," in *IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pp. 29–33, 2019.

[29] S. F. Huq, A. Z. Sadiq, and K. Sakib, "Is developer sentiment related to software bugs: An exploratory study on github commits," in *IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 527–531, 2020.

[30] S. F. Huq, A. Z. Sadiq, and K. Sakib, "Understanding the effect of developer sentiment on fix-inducing changes: An exploratory study on github pull requests," in *26th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 514–521, 2019.

[31] N. Singh and P. Singh, "How do code refactoring activities impact software developers' sentiments? - an empirical investigation into github commits," in *24th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 648–653, 2017.

[32] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on github," in *IEEE/ACM 11th Working Conference on Mining Software Repositories (MSR)*, pp. 348–351, 2014.

[33] M. Ortu, A. Murgia, G. Destefanis, P. Tourani, R. Tonelli, M. Marchesi, and B. Adams, "The emotional side of software developers in jira," in *IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pp. 480–483, 2016.

[34] Z. Huang, *"Replication package of this paper"*, Available at: https://doi.org/10.6084/m9.figshare.14178380.v1. Accessed: Mar, 2021.

[35] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, and F. Ferrucci, "Gender diversity and community smells: Insights from the trenches," *IEEE Software*, vol. 37, no. 1, pp. 10–16, 2020.

[36] M. De Stefano, F. Pecorelli, D. A. Tamburri, F. Palomba, and A. De Lucia, "Splicing community patterns and smells: A preliminary study," in *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW)*, p. 703–710, 2020.

[37] T. Ahammed, M. Asad, and K. Sakib, "Understanding the involvement of developers in missing link community smell: An exploratory study on apache projects," in *8th International Workshop on Quantitative Approaches to Software Quality co-located with 27th Asia-Pacific Software Engineering Conference (QuASoQ@APSEC)*, vol. 2767 of *CEUR Workshop Proceedings*, pp. 64–70, 2020.

[38] R. Jongeling, S. Datta, and A. Serebrenik, "Choosing your weapons: On sentiment analysis tools for software engineering research," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 531–535, 2015.

[39] G. Destefanis, M. Ortu, S. Counsell, S. Swift, M. Marchesi, and R. Tonelli, "Software development: Do good manners matter?," *PeerJ Computer Science*, vol. 2, p. e73, 2016.

[40] I. A. Khan, W. Brinkman, and R. M. Hierons, "Do moods affect programmers' debug performance?," *Cognition Technology & Work*, vol. 13, no. 4, pp. 245–258, 2011.

[41] G. Destefanis, M. Ortu, D. Bowes, M. Marchesi, and R. Tonelli, "On measuring affects of github issues' commenters," in *IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pp. 14–19, 2018.

[42] D. Graziotin, X. Wang, and P. Abrahamsson, "Happy software developers solve problems better: psychological measurements in empirical software engineering," *PeerJ*, vol. 2, p. e289, 2014.

[43] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "Consequences of unhappiness while developing software," in *IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pp. 42–47, 2017.

[44] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "What happens when software developers are (un)happy," *Journal of Systems and Software (JSS)*, vol. 140, pp. 32–47, 2018.

[45] M. R. Islam and M. F. Zibran, "Towards understanding and exploiting developers' emotional variations in software engineering," in *IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 185–192, 2016.

[46] D. Spadini, M. F. Aniche, and A. Bacchelli, "Pydriller: Python framework for mining software repositories," in *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE)*, pp. 908–911, 2018.

[47] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *Journal of the American Society for Information Science and Technology (JASIST)*, vol. 63, no. 1, pp. 163–173, 2012.

[48] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts, "A computational approach to politeness with application to social factors," in *51st Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1: Long Papers, pp. 250–259, 2013.

[49] T. D. Smedt and W. Daelemans, "Pattern for python," *Journal of Machine Learning Research (JMLR)*, vol. 13, no. 66, pp. 2063–2067, 2012.

[50] D. Graziotin, X. Wang, and P. Abrahamsson, "Software developers, moods, emotions, and performance," *IEEE Software*, vol. 31, no. 4, pp. 24–27, 2014.

[51] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," *IEEE Transactions on Software Engineering (TSE)*, vol. 46, no. 11, pp. 1200–1219, 2020.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research (JMLR)*, vol. 12, no. 85, p. 2825–2830, 2011.

[53] F. Palomba, M. Zanoni, F. A. Fontana, A. De Lucia, and R. Oliveto, "Toward a smell-aware bug prediction model," *IEEE Transactions on Software Engineering (TSE)*, vol. 45, no. 2, pp. 194–218, 2019.

[54] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering (TSE)*, vol. 43, no. 1, pp. 1–18, 2017.

[55] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[56] A. Al-Ani and M. Deriche, "Feature selection using a mutual information based measure," in *Object recognition supported by user interaction for service robots*, vol. 4, pp. 82–85, 2002.

[57] A. J. Scott and M. Knott, "A cluster analysis method for grouping means in the analysis of variance," *Biometrics*, vol. 30, no. 3, pp. 507–512, 1974.

[58] R. J. Grissom and J. J. Kim, *Effect sizes for research: A broad practical approach.* Lawrence Erlbaum Associates Publishers, 2005.

[59] X. Yang, H. Yu, G. Fan, and K. Yang, "A differential evolution-based approach for effort-aware just-in-time software defect prediction," in *1st ACM SIGSOFT International Workshop on Representation Learning for Software Engineering and Program Languages co-located with 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (RL+SEPL@ESEC/SIGSOFT FSE)*, p. 13–16, 2020.

[60] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," *IEEE Transactions on Software Engineering (TSE)*, vol. 45, no. 7, pp. 683–711, 2019.

[61] F. Palomba, A. Panichella, A. Zaidman, R. Oliveto, and A. D. Lucia, "The scent of a smell: An extensive comparison between textual and structural smells," *IEEE Transactions on Software Engineering (TSE)*, vol. 44, no. 10, pp. 977–1000, 2018.

[62] H. Hofmann, H. Wickham, and K. Kafadar, "Letter-value plots: Boxplots for large data," *Journal of Computational and Graphical Statistics (JCGS)*, vol. 26, no. 3, pp. 469–477, 2017.

[63] N. Ubayashi, Y. Kamei, and R. Sato, "When and why do software developers face uncertainty?," in *IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, pp. 288–299, 2019.

[64] M. Ortu, M. Marchesi, and R. Tonelli, "Empirical analysis of affect of merged issues on github," in *IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pp. 46–48, 2019.

[65] W. N. Robinson, T. Deng, and Z. Qi, "Developer behavior and sentiment from data mining open source repositories," in *49th Hawaii International Conference on System Sciences (HICSS)*, pp. 3729–3738, 2016.

[66] M. Hozano, A. Garcia, N. Antunes, B. Fonseca, and E. Costa, "Smells are sensitive to developers! on the efficiency of (un)guided customized detection," in *IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pp. 110–120, 2017.

[67] L. Gren, P. Lenberg, and K. Ljungberg, "What software engineering can learn from research on affect in social psychology," in *IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pp. 38–41, 2019.

[68] N. Sae-Lim, S. Hayashi, and M. Saeki, "Revisiting context-based code smells prioritization: On supporting referred context," in *Proceedings of the XP2017 Scientific Workshops*, pp. 1–5, 2017.

[69] N. Sae-Lim, S. Hayashi, and M. Saeki, "Context-based approach to prioritize code smells for prefactoring," *Journal of Software: Evolution and Process (JSEP)*, vol. 30, no. 6, p. e1886, 2018.

[70] N. Novielli, F. Calefato, D. Dongiovanni, D. Girardi, and F. Lanubile, "Can we use se-specific sentiment analysis tools in a cross-platform setting?," in *IEEE/ACM 17th International Conference on Mining Software Repositories (MSR)*, p. 158–168, 2020.

[71] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: How far can we go?," in *IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pp. 94–104, 2018.

[72] P. Tourani, Y. Jiang, and B. Adams, "Monitoring sentiment in open source mailing lists: Exploratory study on the apache ecosystem," in *24th Annual International Conference on Computer Science and Software Engineering (CASCON)*, p. 34–44, 2014.

[73] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen, "Communication in open source software development mailing lists," in *IEEE/ACM 10th Working Conference on Mining Software Repositories (MSR)*, pp. 277–286, 2013.

[74] F. Calefato, F. Lanubile, N. Novielli, and L. Quaranta, "Emtk - the emotion mining toolkit," in *IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pp. 34–37, 2019.

[75] Z. Chen, Y. Cao, X. Lu, Q. Mei, and X. Liu, "Sentimoji: An emoji-powered learning approach for sentiment analysis in software engineering," in *27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE)*, pp. 841–852, 2019.

[76] R. Oliveira, R. de Mello, E. Fernandes, A. Garcia, and C. Lucena, "Collaborative or individual identification of code smells? on the effectiveness of novice and professional developers," *Information and Software Technology (IST)*, vol. 120, p. 106242, 2020.

[77] V. Lenarduzzi, N. Saarimaki, and D. Taibi, "On the diffuseness of code technical debt in java projects of the apache ecosystem," in *IEEE/ACM International Conference on Technical Debt (TechDebt)*, pp. 98–107, 2019.

[78] M. Kuutila, M. V. Mãntyl a, and M. Claes, "Chat activity is a better predictor than chat sentiment on software developers productivity," in *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW)*, p. 553–556, 2020.