FEATURE REDUCTION TECHNIQUES BASED CODE SMELL PREDICTION

By

PRAVIN SINGH YADAV * RAJWANT SINGH RAO **

*-** Department of Computer Science & Information Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur, India.

Date Received: 08/11/2022 Date Revised: 14/11/2022 Date Accepted: 20/11/2022

ABSTRACT

Code Smell refers to the telltale signs of poor code design that leads to software quality issues. Developers require specific methods to measure the complexity of Code Smells in order to resolve the problem quickly. Recent research has examined the problem of predicting Code Smell using various detection methods. However, the accuracy of machine learning-based Code Smell detectors is still at a normal level. One of the main objective of this paper is to assess how well dimensionality reduction methods can predict Code Smells. This paper uses three machine learning techniques with feature reduction techniques, such as Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and Linear Discriminate Analysis (LDA). Ten-fold cross-validation is used to ensure that the model is well-trained. Datasets are balanced using the Synthetic Minority Oversampling Technique (SMOTE) to ensure an equal number of classes in each dataset. The experimental result concluded that the AdaBoost method with LDA performs better in both the Long Parameter List and Switch Statement datasets, with an accuracy of 92.72% and 91.24%, respectively.

Keywords- Code Smell, Code Smell Prediction, Data Balancing, Feature Selection, Machine Learning Techniques.

INTRODUCTION

Software with a bad Code Smell has a structural flaw that could be difficult to fix in the future. There is a chance that it will slow down the operations or cause the software to crash. It is crucial to anticipate Code Smells early on. Code Smells can be predicted using the refactoring strategy's applications (Yamashita & Moonen, 2012). Code Smells can be reduced, software quality and maintenance can be improved through the refactoring process. 22 different types of code smells have been defined. (Fowler, n.d.).

Due to the code's size and complexity, maintaining software on a massive scale is a formidable challenge. Software maintenance costs are more than 80% of the entire development price (Catolino et al., 2020).

There are some limitations to the Code Smell detectors. Code Smells are typically challenging to pin down since the definitions are subject to interpretation (Paiva et al., 2017).

This paper primarily focuses on measuring the efficacy of feature-reduction techniques for classifying Code Smells. This research used benchmark datasets from an earlier investigation by Arcelli Fontana et al. (2016); different machine learning (ML) models were used.

The 420 instances are in the Long Parameter List and Switch Statement datasets, which are examples of method-level Code Smells. Software metrics are there in the datasets as features. This paper employs feature reduction methods to improve the accuracy of Code Smell prediction. This paper used three ML methods to identify Code Smells in datasets that employ a dimensionality reduction strategy.

1. Literature Review

Kreimer (2005) suggested a method for detecting two

This paper has objectives related to SDGS

Code Smells (Long Method and Large Class) in two different software applications using a Decision Tree (DT) model. The model's accuracy was entirely satisfactory.

Liu et al. (2021) combined a deep learning strategy with bootstrap ensemble learning to identify four distinct smells in source code. Ten widely used open-source programs were used for testing and employed an automatic detection method to generate a massive collection of training data. The F-measure values for four different types of Code Smell detectors have improved, and the evaluation results demonstrate that the proposed model is significantly improved over prior techniques.

Abdou and Darwish (2018) proposed a comparison study between individual and ensemble learners to forecast software errors. The preprocessing step of Synthetic Minority Oversampling Technique (SMOTE) resampling has been employed to rectify the data imbalance issue. Predicting software flaws has been done using several ensemble learning methods.

Dewangan and Rao (2022) presented five ML classifiers to extract Code Smells from four Code Smell datasets. In addition, the essential features are extracted from each dataset using a feature selection technique. Using the Random Forest (RF) model on the feature envy dataset, and achieved 99.12% accuracy.

Dewangan et al. (2022) discussed effective Code Smell detection and improvements, achieved with the application of Principal component analysis based Logistic regression (PCA_LR), Principal Component Analysis based k-nearest neighbor (PCA_KNN), Principal Component Analysis based Random Forest (PCA_RF), Principal Component Analysis based Decision Tree method (PCA_DT). The Data class obtains 99.97% accuracy with PCA_LR.

Yadav et al. (2021) suggested a Decision Tree (DT) model to identify the Code Smell with hyperparameter tuning, which achieved an accuracy of 97.62% in both the blob class and the data class datasets.

Alazba and Aljamaan (2021) used the gain metrics selection approach and a Stack-Support Vector Machine (SVM) model to get the best possible outcome. The model achieved 88.89% and 92.50% accuracy with the Gaussian Process (GP) classifier for the Switch Statement and Long Parameter List datasets.

Aljamaan (2021) constructed a heterogeneous voting ensemble using DT, Logistic Regression, Support Vector Machine, Multi-Layer Perceptrons, and Stochastic Gradient Descent models. Soft voting was used to compile the predictions from different participants into an overall ensemble prediction. The voting ensemble obtains 91.8% and 87.8% accuracy in predicting the Long Parameter List and Switch Statement datasets, respectively.

Dewangan et al. (2021) employed six ML approaches with chi-square and Wrapper- Based feature selection techniques. The classifier obtained 100% results using the Logistic Regression for Long Method dataset.

Dewangan et al. (2022) presented a Code Smell prediction approach using ensemble ML. The author utilized five ML and two deep learning techniques and obtained 100% accuracy in the Long Method dataset.

2. Research Methodologies

Classifiers based on ML have been built to identify Code Smells. A solution to the data imbalance problem in training sets has been implemented using the SMOTE resampling method. The literature makes use of a wide variety of dimensionality reduction methods. Figure 1 shows and explains the flow of the proposed experiment.

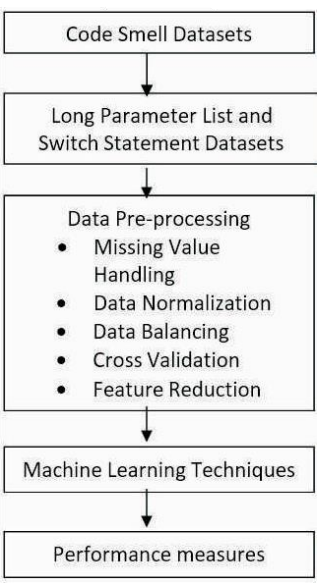

Figure 1. Workflow of Proposed System

The primary objective of this research is to optimize feature reduction for improved classifier performance.

2.1 Reference Datasets

This investigation makes use of benchmark data used by (Arcelli Fontana et al., 2016). The 76 Java systems included in the datasets cover a wide range of classes, packages, and different application fields. The 76 systems are used to calculate a huge number of object-oriented metrics. The definition of these measures was influenced by software quality factors like complexity, scale, and coupling. In this paper, Long Parameter List and Switch Statement datasets are used, they belong to method-level datasets. As shown in Table 1, each dataset contains 420 instances, of which 140 instances are smelly and 280 instances are non-smelly.

2.2 Long Parameter List

A Long Parameter List is an example of a Code Smell at the level of a method. As in this method, an excessively high number of parameters would make performing the function more challenging. This smell belongs to the Bloaters smell group (Alazba & Aljamaan, 2021).

2.3 Switch Statement

A Switch Statement Code Smell is an example of a method-level Code Smell. The smell is specified using a Switch Statement or a chain of if statements. There is a Code Smell which adds a new condition that requires so many other changes. These Code Smells come under object-oriented abusers (Alazba & Aljamaan, 2021).

2.4 Data Preprocessing

This process consists of several data pre-processing steps, such as missing value handling, data normalization, data balancing, cross-validation, and dimension reduction. When dealing with missing data, the mean value imputation method is applied. Min-max normalization is used in datasets. SMOTE is utilized to balance the number of classes in the datasets. To obtain better performance, ten-fold cross-validation is applied. It divides the dataset into ten parts, and for each iteration, one-fold is used as testing data, and the rest of the part is used as training data.

Code Smell	Features used	Number of instances	Number of Smelly instances	Number of non-smelly instances
Long Parameter List dataset	57	420	140	280
Switch Statement dataset	57	420	140	280

Table 1. Datasets Description

2.5 Dimension Reduction

Feature selection and feature reduction techniques come under "dimensionality reduction." In this paper, three feature reduction techniques are utilized in the Code Smell datasets.

- Principal Component Analysis (PCA): PCA is a widely used feature reduction technique. By mapping the original set of features into a smaller set of main components, PCA can reduce the total number of features (Genender-Feltheimer, 2018).
- t- Distributed Stochastic Neighbor Embedding (t-SNE): t-SNE is a rare method in that it can preserve both the local and global structure of the data. It does the same for high-dimensional space, calculating the likelihood of how similar two points are to one another (Genender-Feltheimer, 2018).
- Linear Discriminant Analysis (LDA): LDA is a mathematical method for analyzing different classes of objects or items by combining many data points and applying a function to the combined set (Genender-Feltheimer, 2018).

2.6 ML Technique

Three ML methods are employed in this paper, they are Random Forest, Decision Tree, and Adaptive Boosting.

- Random Forest: An RF is a collection of unpruned classification trees constructed from randomly selected subsets of the training data. The induction procedure chooses features at random. Predictions from an ensemble are combined to form a single forecast (Arcelli Fontana & Zanoni, 2017).
- Decision Tree: A DT is an ML model that maps features and attributes to nodes, decisions, and rules to branches, and outputs to leaves. It mirrors human-level reasoning, making it incredibly simple to gather the facts and generate intelligent conclusions. The objective is to organize all the data into a tree and

process a single result at each leaf (Arcelli Fontana & Zanoni, 2017).

- Adaptive Boosting: Adaptive Boosting, also known as AdaBoost, is an ensemble method used in machine learning. It strengthens a single poor classifier by combining many weak ones (Freund & Schapire,1996).

2.7 *Performance Measures*

This paper tested the Accuracy, Recall, F1-score, and precision using confusion metrics. At first, it uses the abbreviations True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) to indicate a result. Accuracy Recall, F1-score, and precision were defined using Equations 1 to 4.

$$Accuracy\ (A) = \frac{TP + TN}{TP+TN + FP + FN} \quad (1)$$

$$Precision\ (P) = \frac{TP}{TP+FP} \quad (2)$$

$$Recall\ (R) = \frac{TP}{TP+FN} \quad (3)$$

$$F1\text{ - }Score\ (F1) = 2X\frac{PXR}{P+R} \quad (4)$$

3. Result And Discussion

This paper predicts Code Smells using data from two Code Smell datasets, such as Switch Statements and Long Parameter Lists. Tables 2 to 4 show the findings. In this paper, PCA with DT, RF, and AdaBoost are denoted by P_DT, P_RF, and P_AdaBoost. LDA with DT, RF, and AdaBoost are represented by L_DT, L_RF, and L_AdaBoost, respectively. T_DT, T_RF, and T_AdaBoost refer to t-SNE with

Numberof Components	Switch Statement dataset						Long Parameter List dataset					
	L_DT		L_RF		L_AdaBoost		L_DT		L_RF		L_AdaBoost	
	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)
1	87.97	87.93	87.97	87.93	91.24	91.20	91.65	91.71	91.65	91.71	92.72	92.69

Table 2. Result of LDA in Switch Statement and Long Parameter List Datasets

Numberof Components	Switch Statement dataset						Long Parameter List dataset					
	T_DT		T_RF		T_AdaBoost		T_DT		T_RF		T_AdaBoost	
	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)
1	82.12	81.97	80.90	80.79	76.27	76.31	78.39	78.37	78.73	78.46	68.98	68.12
2	82.99	82.93	85.90	85.92	70.29	70.37	80.33	80.85	83.34	83.54	70.20	69.38
3	73.69	74.10	79.21	79.29	67.68	65.40	71.28	71.62	74.47	74.24	61.16	61.28

Table 3. Result of t-SNE in Switch Statement and Long Parameter List Datasets

Numberof Components	Switch Statement dataset						Long Parameter List dataset					
	P_DT		P_RF		P_AdaBoost		P_DT		P_RF		P_AdaBoost	
	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)	Accuracy (%)	F1-Score (%)
1	59.12	59.87	59.12	58.87	62.86	66.60	58.87	58.46	58.87	58.46	57.30	57.88
2	70.60	71.25	75.58	76.83	74.91	75.32	63.31	63.97	69.14	69.44	57.47	58.24
3	81.25	81.24	85.04	85.35	81.77	82.21	66.68	67.05	71.96	72.19	64.38	65.33
4	80.56	80.39	84.52	84.75	79.88	80.09	69.65	70.34	75.33	75.51	65.96	66.13
5	80.94	81.28	85.22	85.48	81.07	81.26	79.43	79.75	84.91	85.38	76.43	76.89
6	84.02	84.30	87.62	87.75	81.42	81.49	78.71	79.05	86.16	86.57	75.20	75.60
7	83.33	83.34	87.63	87.82	83.31	83.13	81.91	82.33	86.51	87.01	78.19	78.64
8	83.33	83.56	88.14	88.20	86.41	86.52	82.97	83.42	87.95	88.19	80.51	81.10
9	82.81	82.97	87.79	87.84	84.68	84.83	81.91	82.40	87.94	88.15	79.25	79.54
10	81.43	81.50	87.62	87.78	85.37	85.42	83.33	83.74	88.11	88.40	81.21	81.44
11	82.30	82.26	88.14	88.08	84.18	84.28	82.61	83.02	87.58	87.82	82.09	82.53
12	82.30	82.62	88.31	88.20	85.22	84.98	84.93	85.18	88.30	88.56	82.25	82.75
13	81.78	82.15	88.48	88.53	85.38	85.48	84.92	85.05	88.29	88.48	82.45	82.91
With all Components	80.22	80.17	89.69	89.81	85.38	85.73	81.38	81.62	91.30	91.05	89.72	89.72

Table 4. Result of PCA in Switch Statement and Long Parameter List Datasets

DT, RF, and AdaBoost, respectively. PCA is utilized with different components from 1 to 13 in this paper. In the case of a t-SNE, the maximum number of components is three; it is utilized for three to one component. In the case of LDA, it is utilized only one component because the dataset belongs to a binary class.

P_RF model achieved the highest accuracy of 89.69% and 91.30% with all components in the Switch Statement dataset and Long Parameter List dataset, respectively. In t-SNE, two components acquire the highest accuracy of 85.90% and 83.34% in the Switch Statement and Long Parameter List datasets. In LDA highest accuracy obtained are 91.24% in the Switch Statement dataset and 92.72% in the Long Parameter List dataset.

Table 5 shows a comparative study of the results with previous papers. This approach performs better in the Switch Statement dataset, with the highest accuracy of 91.24%. In the Long Parameter List dataset, the approach performed better than that (Alazba & Aljamaan, 2021). In comparison, (Dewangan et al., 2022) got their result with 55 components, and the proposed system got its result with one component. Comparatively, this paper gives better results.

Conclusion

In this paper, three feature reduction techniques, PCA, LDA, and t-SNE, are utilized with three ML algorithms. The model is adequately trained using ten-fold cross-validation. This paper focuses on two method-level Code Smells that are publicly available. In order to fairly divide up the various classes, the SMOTE Data Balancing Method is considered. Improving performance while decreasing required computing time is the primary focus of this paper. It is found that the AdaBoost method with LDA achieves the highest accuracy of 92.72% and 91.24% for the Long Parameter List and Switch Statement Datasets, respectively.

In the future, researchers will use parameter optimization techniquesto properly tune the model and apply different ML algorithms to achieve better performance in Code Smell detection.

References

[1]. Abdou, A. S., & Darwish, N. R. (2018). Early Prediction Of Software Defect Using Ensemble Learning: A Comparative Study. *International Journal Of Computer Applications*, 179(46), 29-40.

[2]. Alazba, A., & Aljamaan, H. (2021). Code Smell Detection Using Feature Selection And Stacking Ensemble: An Empirical Investigation. *Information And Software Technology*, 138, 106648. Https://Doi.Org/10.1016/J.INFSOF.2021.106648

[3]. Aljamaan, H. (2021). Voting Heterogeneous Ensemble For Code Smell Detection. *Proceedings - 20th IEEE International Conference On Machine Learning And Applications*, 897–902. Https://Doi.Org/10.1109/ICMLA52953.2021.00148

[4]. Arcelli Fontana, F., Mäntylä, M. V., Zanoni, M., & Marino, A. (2016). Comparing And Experimenting Machine Learning Techniques For Code Smell Detection. *Empirical Software Engineering*, 21(3), 1143-1191. Https://Doi.Org/10.1007/S10664-015-9378-4/TABLES/24

[5]. ontana, F. A., & Zanoni, M. (2017). Code Smell Severity Classification Using Machine Learning Techniques. *Knowledge-Based Systems*, 128, 43-58. Https://Doi.Org/10.1016/J.KNOSYS.2017.04.014

[6]. Catolino, G., Palomba, F., Fontana, F. A., De Lucia, A., Zaidman, A., & Ferrucci, F. (2020). Improving Change Prediction Models With Code Smell-Related Information.

Year of publication	Reference	Datasets					
		Switch Statement Dataset			Long Parameter List Dataset		
		Approach	Number of components/ features	Accuracy (%)	Approach	Number of components/ features	Accuracy (%)
2021	(Alazba & Aljamaan, 2021)	GP	25	88.89	GP	22	92.50
2022	(Dewangan et al., 2022)	PCA_KNN	9	85.72	PCA_LR and PCA_RF	55	94.05
	Proposed approach	L_AdaBoost	1	91.24	L_AdaBoost	1	92.72

Table 5. Comparison of Result with Previous Work

Empirical Software Engineering, 25(1), 49-95. Https://Doi.Org/10.1007/S10664-019-09739-0/FIGURES/3

[7]. Dewangan, S., & Rao, R. S. (2022). Code Smell Detection Using Classification Approaches. In *Intelligent Systems* (Pp. 257-266). Springer, Singapore. Https://Doi.Org/10.1007/978-981-19-0901-6_25

[8]. Dewangan, S., Rao, R. S., Mishra, A., & Gupta, M. (2021). A Novel Approach For Code Smell Detection: An Empirical Study. *IEEE Access*, 9, 162869-162883. Https://Doi.Org/10.1109/ACCESS.2021.3133810

[9]. Dewangan, S., Rao, R. S., Mishra, A., & Gupta, M. (2022). Code Smell Detection Using Ensemble Machine Learning Algorithms. *Applied Sciences*, 12(20), 10321. Https://Doi.Org/10.3390/APP122010321

[10]. Dewangan, S., Rao, R. S., & Yadav, P. S. (2022, July). Dimensionally Reduction Based Machine Learning Approaches For Code Smells Detection. In *2022 International Conference On Intelligent Controller And Computing For Smart Power (ICICCSP)* (Pp. 1-4). IEEE. Https://Doi.Org/10.1109/ICICCSP53532.2022.9862030

[11]. Fowler, M. (N.D.). *Refactoring: Improving The Design Of Existing Code*. Https://Www.Amazon.In/Refactoring-Improving-Existing-Addison-Wesley-Signature/Dp/0134757599

[12]. Freund, Y., & Schapire, R. E. (1996). Experiments With A New Boosting Algorithm. *Machine Learning: Proceedings Of The Thirteenth International Conference*, 1-9.

[13]. Genender-Feltheimer, A. (2018). Visualizing High Dimensional And Big Data. *Procedia Computer Science*, 140, 112-121. Https://Doi.Org/10.1016/J.PROCS.2018.10.308

[14]. Kreimer, J. (2005). Adaptive Detection Of Design Flaws. *Electronic Notes In Theoretical Computer Science*, 141(4), 117-136. Https://Doi.Org/10.1016/J.ENTCS.2005.02.059

[15]. Liu, H., Jin, J., Xu, Z., Zou, Y., Bu, Y., & Zhang, L. (2019). Deep Learning Based Code Smell Detection. *IEEE Transactions On Software Engineering*, 47(9), 1811-1837. Https://Doi.Org/10.1109/TSE.2019.2936376

[16]. Paiva, T., Damasceno, A., Figueiredo, E., & Sant'Anna, C. (2017). On The Evaluation Of Code Smells And Detection Tools. *Journal Of Software Engineering Research And Development*, 5(1), 1–28. Https://Doi.Org/10.1186/S40411-017-0041-1

[17]. Yadav, P. S., Dewangan, S., & Rao, R. S. (2021, December). Extraction Of Prediction Rules Of Code Smell Using Decision Tree Algorithm. In *2021 10th International Conference On Internet Of Everything, Microwave Engineering, Communication And Networks (IEMECON)* (Pp. 1-5). IEEE. Https://Doi.Org/10.1109/IEMECON53809.2021.9689174

[18]. Yamashita, A., & Moonen, L. (2012, September). Do Code Smells Reflect Important Maintainability Aspects?. In *2012 28th IEEE International Conference On Software Maintenance (ICSM)* (Pp. 306-315). IEEE. Https://Doi.Org/10.1109/ICSM.2012.6405287

ABOUT THE AUTHORS

D