PREDICTION OF DEFECTS WITH CHANGES IN SOFTWARE

JINAL GUPTA [1*], DR. ANSHU KHURANA [2], NITISH UPPAL [3]

[1 2 3]Department of Artificial Intelligence and Data Science, Maharaja Agrasen Institute of Technology, Rohini, Delhi
[*] Corresponding Author. E-mail: jinalbirla@gmail.com

Abstract*: Software defect prediction proactively highlights critically vulnerable elements for a software system. Recent attempts in this domain have tried various machine learning techniques to automate and improve software defect prediction. It has been observed that recent research focusses on specific types of software and hence the solution is not suitable for all kind of software products. This paper proposes a generic approach based on Logistic Regression, Decision Trees, Random Forests, Naïve Bayes, Support Vector Machine for software defect prediction. Proposed approach is applicable on all type of software products and the results shows defect occurrence are not directly correlated with the quantity of modifications.*

Keywords: Defect Prediction, Machine Learning Techniques, Software defect prediction Metrices, Ensemble Technique, Attribute correlation

I. INTRODUCTION

The need for producing affordable, maintainable software without sacrificing quality has increased due to the software systems' growing dependence. The likelihood of the software having a flaw is therefore very high. If errors are identified later in the software development process, the associated costs for their prompt resolution significantly increases. Software prediction models can be applied to initial phases of the software development life cycles. Additionally, utilizing them cuts down on project costs, time, and effort spent on testing and maintenance, which boosts the software's quality. We can work to strengthen the software's weaker components by identifying the defective portions of those components. Thus, it is possible to create high-quality software with inexpensive development and maintenance costs.

Researchers have developed models that can predict these issues precisely and rapidly. These methods include the use of cutting-edge techniques like machine learning and data analysis. One of the efficient methods in this area may be ensemble techniques. They function by merging the results of various models, which increases their accuracy and dependability. It's similar to consulting with several experts before making a decision. This study focuses on investigating how ensemble techniques might be applied to improve software defect prediction. By doing this, we intend to raise the standard of software, lessen the effort required to address problems, and enhance the functionality of software systems as a whole.

This research paper's major goal is to examine how ensemble techniques can be used to forecast software flaws and to comprehend the advantages they have over more conventional single-model methods. We studied their attributes individually. The following are the objectives of the study:

1. Review and analyze the available investigation on ensemble strategies for predicting software defects, including their methodologies, algorithms, and performance measures.
2. To increase the precision and dependability of predictions, propose and create a defect prediction framework that integrates various base models, such as decision trees, support vector machines, or neural networks.
3. Run thorough tests and assessments on standard datasets to compare ensemble techniques' performance to that of individual models and other cutting-edge fault prediction techniques.
4. Highlight the strengths, weaknesses, and practical consequences of employing ensemble approaches for software defect prediction as you analyze and interpret the data.
5. After analyzing the weaknesses and the faults that were found while ensemble technique, we can now correct them by studying the correlation between the attributes and how they are dependable on each other.

The outcomes will aid the awaited testers in identifying the most effective evolutionary algorithm. We examined and inspected the outcomes of five data vaults using six different machine learning models. We further correlated the attributes individually and saw their dependencies and how they affect the defect in the software.

The information in the paper is arranged as follows: Section 2 of the linked effort contains investigations and related literatures. The proposed algorithm that will be used is described in Section 3 along with the calculated experimental findings. In Section 4, the findings were calculated, analyzed and drawn. The summary of the work done in the study is presented in Section 5. In section 6 we discussed the future scope of the research.

II. LITERATURE REVIEW

Software metrics are essential for managing projects, especially when dealing with new technologies like object-oriented design. There are some studies that introduce a set of metrics tailored for object-oriented systems [6][7], grounded in measurement theory and informed by experienced developers. The suggested metrices undergo a formal evaluation based on predefined criteria.

As object-oriented methods gain popularity in software development, researchers are working on metrics to develop new models and improve the old practices. An analysis of metrics by Chidamber and Kemerer [7] shows their usefulness for managers in industries. Empirical data supports these metrics, underscoring their significant impact on the ingenuity, modify effort, and creation effort in object-oriented systems.

The coupling dependency metric (CDM) [5] proves successful in design quality. When applied to case studies involving COBOL, C, C++, and Java systems, CDM outperforms other metrics in anticipating run-time failures and gauging maintenance requirements. This suggests that coupling metrics, like CDM, can reliably predict interaction levels in software products.

AUC, Accuracy, Precision, Recall and Mean are just a few of the different metrics that have been employed in numerous studies to predict the presence of faulty classes in software systems. These performance measures evaluate the connection between object-oriented metrics and flawed classes using statistical methods and classification algorithms. Observers have employed various methods, including Logistic Regression (LR), Naïve Bayes, Random Forest, Decision Trees, Artificial Neural Networks (ANN), Support Vector Machines (SVM), Bayesian approaches, and Artificial Immune Recognition System [(1)-(3)], to develop predictive models and evaluate their efficacy.

In study [3], the object-oriented metrics is employed in the Mozilla dataset in conjunction with LR gave good results in predicting defective classes. In other study, Linear Regression was compared with three machine learning methods (Naive Bayes, and Random Forest) to categorize object-oriented metrics based on the severity of defects. The investigation showed that models created for high severity levels had reduced prediction accuracy, and ML approach performance metrics were generally low.

Other research [2] evaluated defect prediction models using various datasets, including NASA KC1 and Java Telecom, employing logistic regression, decision trees, ANN, SVM, and Multi Objective PSO approaches. The results highlighted the potential of neural networks and Bayesian methods by demonstrating improved performance with various models and strategies. Additionally, it was shown that decision trees (C4.5) produced greater accuracy when decision tree approaches were used together with neural networks, logistic regression, and SVM.

Another research [1] assessed datasets utilizing machine learning algorithms and 10-fold cross-validation methods. These studies demonstrate the efficacy of various methods and models in this field and jointly advance understanding of software fault prediction.

Software metrics, crucial for project management, are being developed for object-oriented systems, evaluated against established criteria. Object-oriented methods are gaining popularity, with metrics by Chidamber and Kemerer proving useful for managers and impacting productivity in empirical studies. The Coupling Dependency Metric (CDM) successfully predicts run-time failures in diverse systems, indicating its reliability in assessing interaction levels.

III. METHOD

3.1 Research Question for Data

RQ1: What are the machine learning model does SDP uses?

Despite the fact that there are numerous machine learning algorithms, five are chosen for this paper: SVM, NB, DT, LR, and RF.

RQ2: What data sources are employed for SDP?

We employed five different Android software application packages in this paper: Telephony, Gallery, Email, Contacts, and Bluetooth.

RQ3: What performance metrices are utilized in SDP?

In this research, we employ diverse object-oriented measures, encompassing Tang et al. measurements, Chidamber & Kemerer indicators, Henderson Sellers indicators, Martin's indicators, and the QMOOD indicator suite. Subsequently, these measures are utilized as features, and we undertake a reduction process for optimization purposes.

RQ4: How can the effectiveness of evolutionary approaches be evaluated using machine learning models?

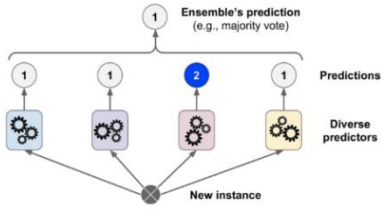

Fig.1: Working of Ensemble technique

Using the Ensemble approach, we validated the results in this work (As shown in Fig1 above).

RQ5: Are there any additional models or techniques used to find the software defect?
Yes, there is one more technique that is by using the dependencies of different attributes from the dataset to visualize how they affect the change in the software and hence create a defect.

3.2.1 Defect prediction (Algorithms)

The actions taken to apply the algorithm to the dataset for Android are as follows:

1. Collect data sets from Android software archives.
2. Calculate all classification techniques' precision, Recall, Accuracy, and F-1 scores using all available features.
3. The scores for Precision, Recall, F-1, and Accuracy obtained from various classification techniques are included in TABLE l, TABLE ll, TABLE lll, and TABLE lV, along with a comparison between them and the evolutionary algorithm Ensemble Technique. The data from TABLE I and II are represented by the precision graph in Fig. 1.
4. Repetition of step 3 is required for the remaining Android datasets.
5. Obtain the ensemble technique's precision, accuracy, F-1 score, and recall values.
6. Compare all the values using graphs and get the best fit model.

3.2.1 Defect prediction (Correlation between different attributes) (The method is explained in Fig.2 given below)

The description of the method shown in Fig.2 is explained as follows:

1. To propose a general method, we perform various visualizing techniques amongst the various attributes which does not depend on the type of software.
2. We take in consideration the following attributes:
   i. Number defects
   ii. Number of changes
   iii. Number of Insertions
   iv. Number of Deletions
   v. Defect count
3. Now investigating the influence of the number of changes on the defect count to find the correlation between them and hence predicting the output on this basis.
4. Further investigating the attributes which might have influenced the number of changes to decrease and increase in the software.

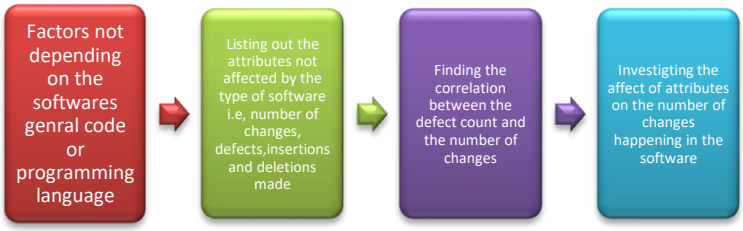

Fig.2: Working for the technique selecting special features to find the reasons for the defect in the software

IV. COMPARE THE RESULTS

4.1.1 Defect Prediction using ensemble technique

We used the ensemble technique to validate the findings.

Experimental Findings:

Precision is calculated as the effectiveness of all datasets for classification procedures in TABLE I. The determined Recall value for each type of dataset is shown in TABLE II. The F-1 scores for each dataset are calculated in TABLE III using various approaches. Accuracy is determined for each dataset in TABLE IV.

Figure 1 displays a graph generated from TABLE I for each of the created models. The table's investigation led us to the following conclusions:

I. In comparison to standalone classification models, RF, DT, SVM, and NB yielded the most significant outcomes. The utilization of ensemble techniques is found to enhance precision.

II. It also demonstrates that the model can produce the best results when employing the voting classifier Ensemble.

Graphs for each model are taken from TABLE II shown in Fig. 2. Following a review of the table, we draw the following conclusions:

I. When compared to classification models alone, recall performed best for DT, RF, and NB. It is concluded that precision can be increased by employing the ensemble technique.

Here is the plot for each model using TABLE III shown in Fig. 3. Following a review of the table, we draw the following conclusions:

I. Compared to conventional classification models, the F-1 score produced the greatest results for DT, RF, and LR. It is concluded that precision can be increased by employing the ensemble technique.

A graph plotted using TABLE IV for all created models is shown in Fig. 4. Following a review of the table, we draw the following conclusions:

I. When compared to traditional classification models, accuracy produced the greatest results with DT, RF, and NB. It is concluded that precision can be increased by employing the ensemble technique.

TABLE I: Precision of Datasets using Ensemble Techniques and Classification

	LR	DF	RF	SVM	NB	ENSEMBLE
Bluetooth	0.5	1	0.8333	0	1	0.833333333
Contacts	1	1	1	1	1	1
Email	0.9375	1	1	0.3125	1	1
Gallery	0.8438	1	1	0.1563	1	1
Telephony	0.9574	1	1	1	1	1

TABLE II: Recall Value of Datasets using Classification and Ensemble Techniques

	LR	DF	RF	SVM	NB	ENSEMBLE
Bluetooth	1	1	1	0	1	1
Contacts	1	1	1	1	1	1
Email	1	1	1	1	0.71111	1
Gallery	1	1	1	1	0.84211	1
Telephony	0.9	1	1	0.6912	0.94	0.979166667

TABLE III: F-1 scores of Datasets using Classification and Ensemble Techniques

	LR	DF	RF	SVM	NB	ENSEMBLE
Bluetooth	0.66667	1	0.909090909	0	1	0.909090909
Contacts	1	1	1	0.711111111	1	1
Email	0.96774	1	1	0.476190476	0.83117	1
Gallery	0.91525	1	1	0.27027027	0.91429	1
Telephony	0.92784	1	1	0.817391304	0.96907	0.989473684

TABLE IV: Accuracy of Datasets using Classification and Ensemble Techniques

	LR	DF	RF	SVM	NB	ENSEMBLE
Bluetooth	0.8636	1	0.954545455	0.727272727	1	0.954545455
Contacts	1	1	1	0.793650794	1	1
Email	0.9859	1	1	0.845070423	0.9085	1
Gallery	0.9558	1	1	0.761061947	0.9469	1
Telephony	0.9744	1	1	0.876923077	0.9846	1

Fig. 3. Graph for computed Precision as performance measures

Fig. 4. Graph for computed Recall as performance measures

Fig. 5. Graph for calculated F-1 score as performance measures

Fig. 6. Graph for calculated Accuracy as performance measures

Following is a summary of the conclusions reached from the above findings:

I. Precision was estimated and analyzed using various machine learning techniques by taking into account TABLE

II. The calculation of Precision using different machine learning models exhibited optimal performance across diverse datasets. For instances, the Bluetooth dataset achieved superior results with the DT and NB Techniques, contact dataset, which performed best using all classification models and Ensemble method, Email dataset, which performed best using all classification models and Ensemble method, and Gallery dataset, which performed best using all classification models.

III. By taking into account TABLE II, the recall computed across various machine learning models, demonstrated optimal performance on distinct datasets. Specifically, the NB technique yielded the best results for the Bluetooth dataset, while the Contacts dataset performed most effectively with DT, LR, RF, and Ensemble Technique. In the case of the Email dataset, superior results were achieved with DT, RF, and Ensemble Technique, whereas the Telephony dataset excelled across all models, except for LR and SVM.

4.1.2 Finding the correlation between the matrices

As for the results we obtained from 4.1.1 we can see that all the models have accurately predicted the defect which shows that the data is biased or either imbalanced and predicts the defect with the slightest of the errors.

As we now know that the data, we are using maybe having an incline towards one of the specific groups so we will now find the dependencies or the relation between the metrices.

We have visualized the relation between the matrices using scatter plots. The Visualization has taken place in between the following attributes:

1) Defect-Count vs Total Number of changes
2)The count of insertions vs Total Number of changes
3)Number of deletions vs Total number of changes
4)Defect-Count vs Number of Deletions
5)Defect-Count vs Number of Insertions

The visualizations are as follows:

1) Defect-Count vs Total Number of changes

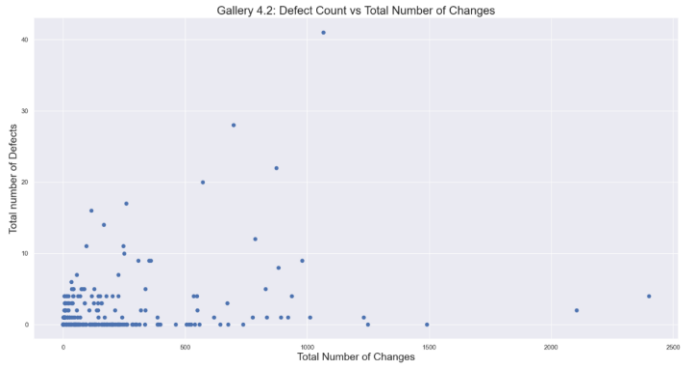

Fig. 7. Scatter plot showing the relation between the Defect Count and the Total Number of changes for Gallery version 4.2 dataset

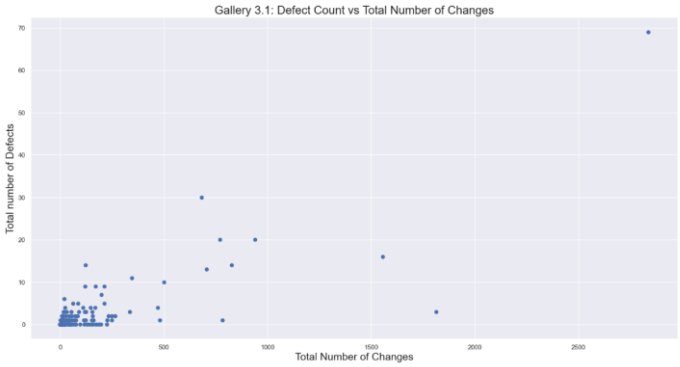

Fig. 8. Scatter plot showing the relation between the Defect Count and the Total Number of changes for Gallery version 3.1 dataset

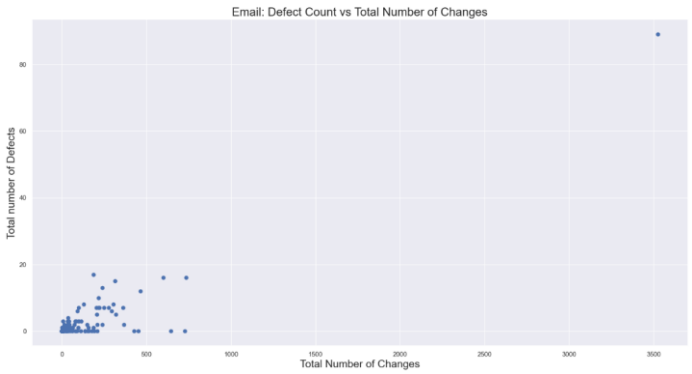

Fig. 9. Scatter plot showing the relation between the Defect Count and the Total Number of changes for email dataset

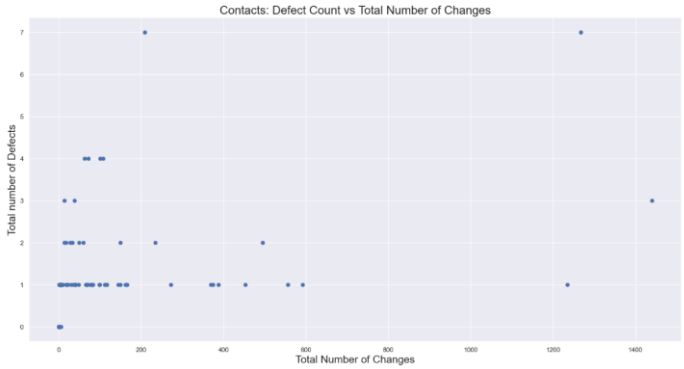

Fig. 10. Scatter plot showing the relation between the Defect Count and the Total Number of changes for contact dataset

I. The relationship between the number of changes and the occurrence of defects is not strictly proportional. While there is an initial trend of increasing defects with the number of changes, this trend does not persist beyond a certain point, typically around an average of 1000 changes.

II. After reaching this threshold, the number of defects remains relatively constant, indicating that additional changes beyond this level do not lead to a significant increase in defects. This suggests that the correlation between the two variables is not direct and may be influenced by additional factors or attributes. [Fig.7] [Fig.8] [Fig.9] [Fig.10]

III. Further investigation into these contributing factors is needed to better understand the dynamics of defect occurrence in relation to changes.

2) Number of insertions vs Total Number of changes

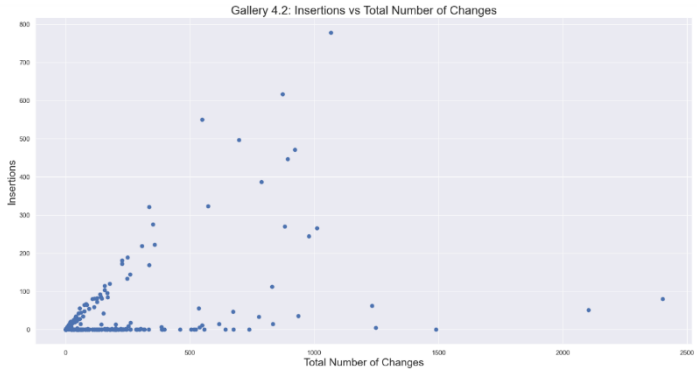

Fig. 11. Graph showing the association between the number of insertions and the Total Number of changes for Gallery version 4.2 dataset

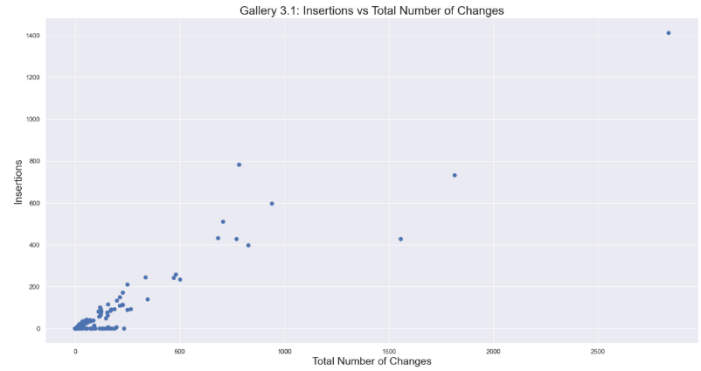

Fig. 12. Scatter plot showing the association between the number of insertions and the Total Number of changes for Gallery version 3.1 dataset

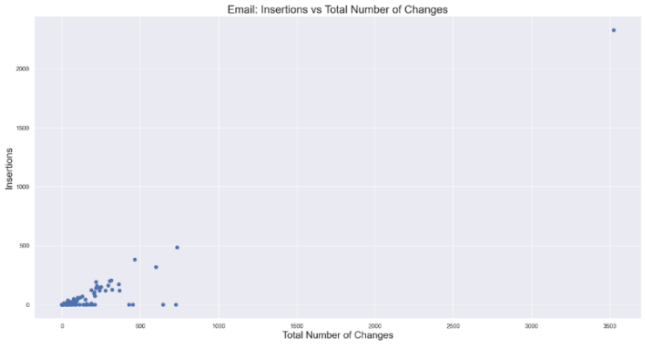

Fig.13. Scatter plot having the association between the number of deletions and the total number of changes

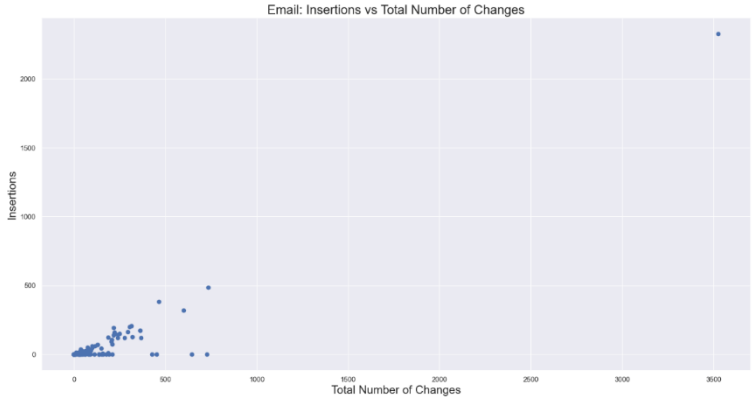

Fig. 14. Scatter plot showing the association between the number of insertions and the Total Number of changes email dataset

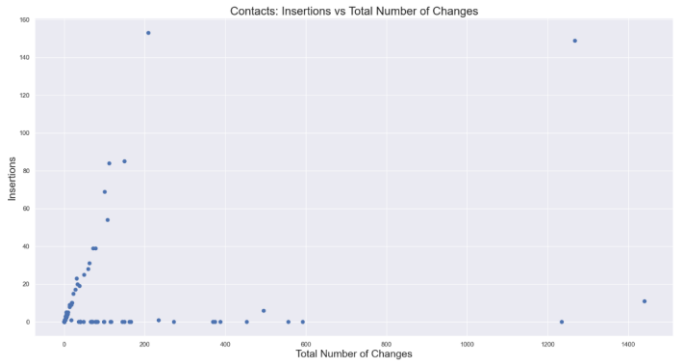

Fig. 15. Scatter plot showing the association between the number of insertions and the Total Number of changes contacts dataset

I. Based on the presented findings, it is evident that an increase in the number of insertions does not result in a notable escalation in the software's modification rate. Consequently, it can be concluded that the frequency of changes remains unaffected by the quantity of insertions made. [Fig.11] [Fig.12] [Fig.13] [Fig.14]

3) Number of deletions vs Total number of changes

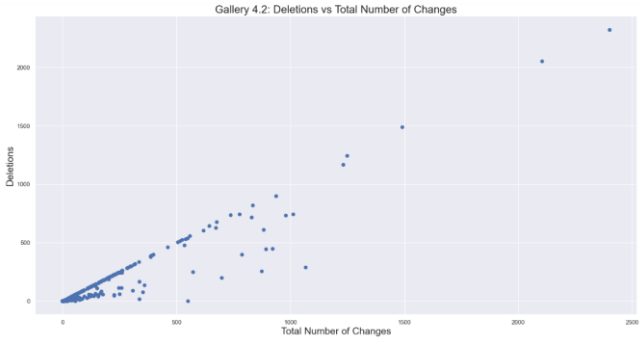

Fig. 16. Scatter plot showing the association between the number of deletions and the Total Number of changes for gallery version 4.2 dataset

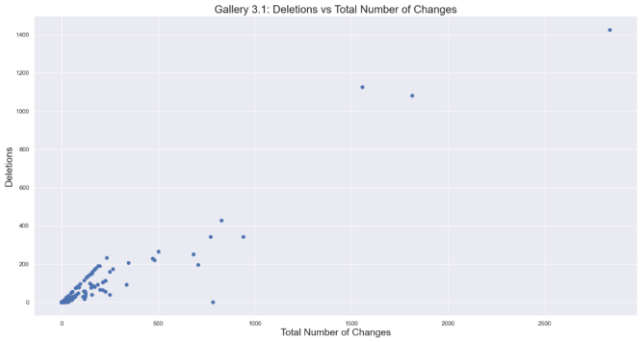

Fig. 17. Scatter plot showing the association between the number of deletions and the Total Number of changes for gallery version 3.1 dataset

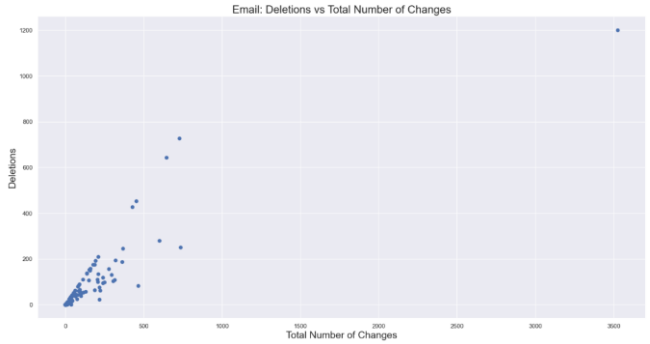

Fig. 18. Scatter plot showing the association between the number of deletions and the Total Number of changes for email dataset

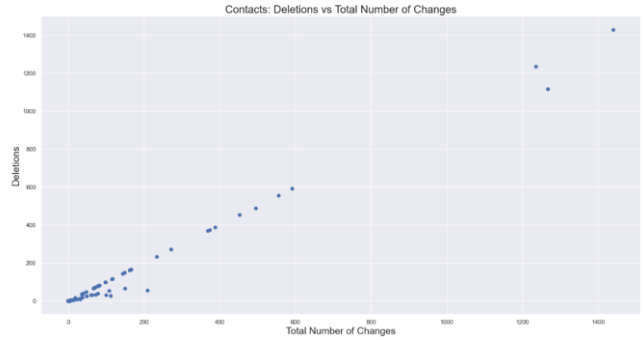

Fig. 19. Scatter plot showing the association between the number of deletions and the Total Number of changes for contacts dataset

I. From the above findings, it is evident that an increase in the number of deletions leads to a rise in the software modification rate. Consequently, it can be concluded that the frequency of changes is affected by the number of deletions happening in the software. [Fig.16] [Fig. 17] [Fig.18] [Fig.19]

4) Defect-Count vs Number of Deletions

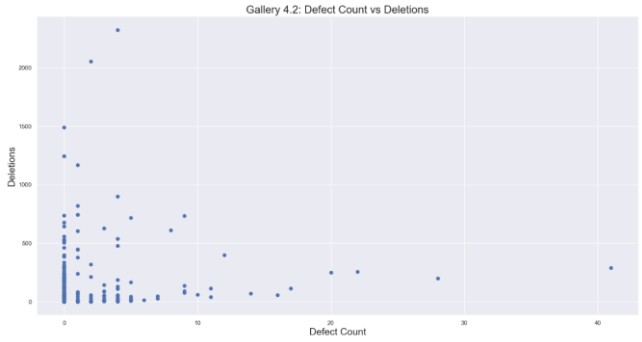

Fig. 20. Scatter plot showing the association between Defect-count and the Total Number of deletions for gallery version 4.2 dataset

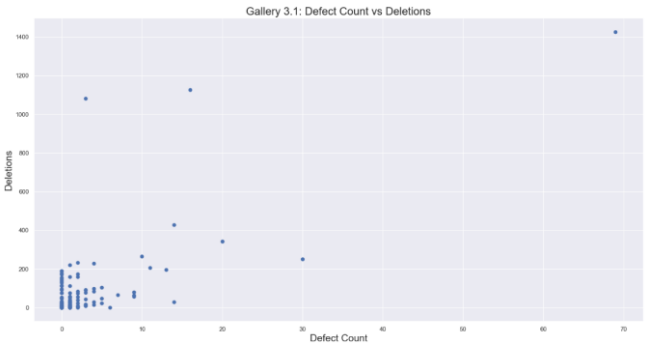

Fig. 21. Scatter plot showing the association between Defect-count and the Total Number of deletions for gallery version 3.1 dataset

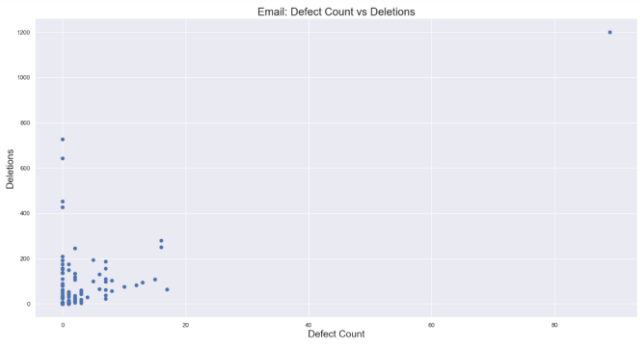

Fig. 22. Scatter plot showing the association between Defect-count and the Total Number of deletions for email dataset

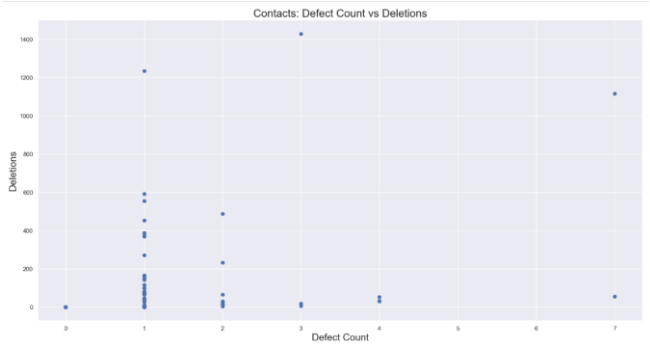

Fig. 23. Scatter plot showing the association between Defect-count and the Total Number of deletions for contacts dataset

I. Based on the presented findings, we can observe that the even after increasing the count of deletions does not surpass an average of 30 defects. Hence it can be concluded that the increase in the number of deletions does not escalates the number of defects in the software design. [Fig.20] [Fig.21] [Fig.22] [Fig.23]

5) Defect-Count vs Number of Insertions

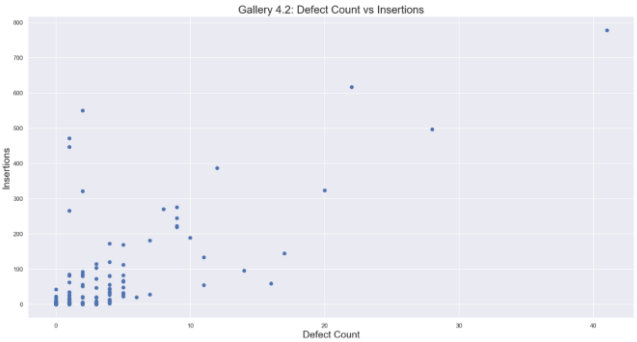

Fig. 24. Scatter plot showing the association between Defect-count and the Total Number of insertions for gallery version 4.2 dataset

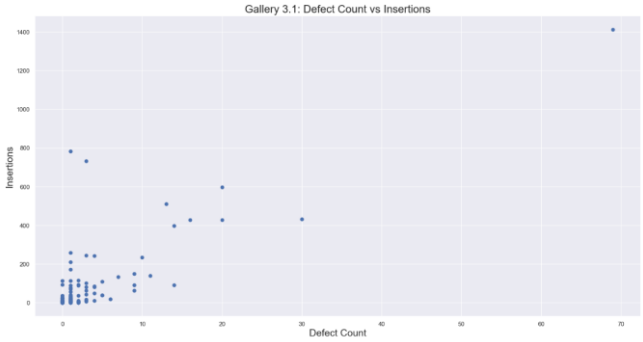

Fig. 25. Scatter plot showing the association between Defect-count and the Total Number of insertions for gallery version 3.1 dataset

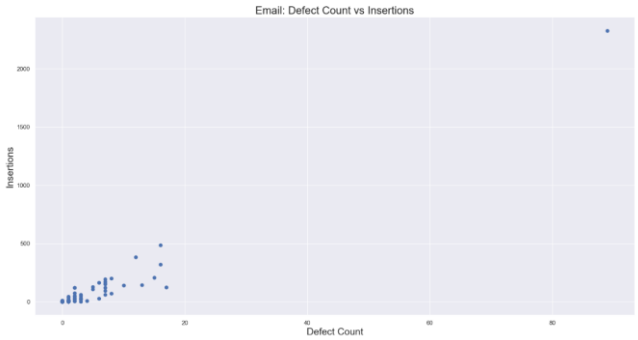

Fig. 26. Scatter plot showing the association between Defect-count and the Total Number of insertions for email dataset

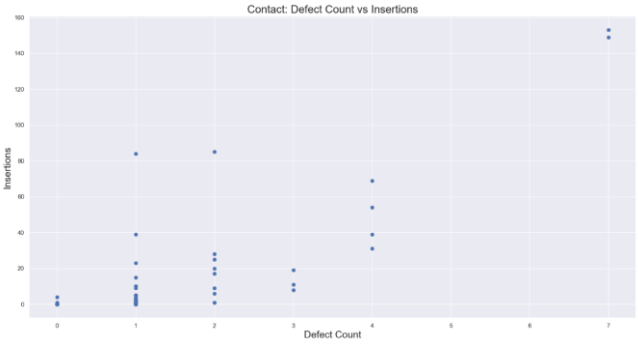

Fig. 27. Scatter plot showing the association between Defect-count and the Total Number of insertions for contacts dataset

I. The relationship between the number of changes and the occurrence of defects in software is complex and not strictly proportional. While there is an initial trend of increasing defects with the number of changes, this trend plateaus after a certain point, typically around an average of 1000 changes.

II. Beyond this threshold, the number of defects remains relatively constant, suggesting that additional changes do not significantly impact defect occurrence.

III. Moreover, the statements indicate that the frequency of changes is not significantly influenced by the number of insertions or deletions in the software.

IV. It is evident from the visualization that defect occurrence is not directly correlated with the number of modifications and may be influenced by additional factors or attributes that require further investigation. [Fig.24] [Fig.25] [Fig.26] [Fig.27]

V. CONCLUSION

Using the Ensemble technique, we were able to provide useful results for future research. The results indicate that applying the Ensemble Technique, the metrics provided over 100% accuracy, indicating that our data is either biased or imbalanced. For further investigation the relationship between the number of changes and the occurrence of defects in software were compared which were found to be complex and not strictly proportional. While there is an initial trend of increasing defects with the number of changes, this trend plateaus after a certain point, typically around an average of 1000 changes (according to the software dataset taken in consideration which can be different for different software). Beyond this threshold, the number of defects remains relatively constant, suggesting that additional changes do not significantly

impact defect occurrence. Moreover, the statements indicate that the frequency of changes is not significantly influenced by the number of insertions or deletions in the software. It is evident that the occurrence of defects is not directly correlated with the number of modifications and may be influenced by additional factors or attributes that require further investigation.

Our research used different prediction techniques for our studies, but we found some issues with our data accuracy, possibly due to bias. When looking at how software changes relate to defects, we noticed that defects increased with more changes, but only up to the threshold the defect occurrence stayed constant. Interestingly, the number of defects wasn't influenced by the new addition or removal of the software components. This complexity suggests there are other factors affecting the number of defects such as the change in the functionality of the software which can be further be taken in consideration while using different techniques.

VI. FUTURE SCOPE

There is room for more learning because the dataset used for the article was biased and unbalanced. For improved outcomes, we can also use the transfer learning technique. The future scope may also include studying about different metrices available in the dataset for evaluation which play a role in shaping defect dynamics, contributing to a more comprehensive understanding of software development challenges and solutions.

VII. REFERENCES

[1] Malhotra, R., & Khurana, A. (2017). Analysis of evolutionary algorithms to improve software defect prediction. 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO).

[2] Ruchika Malhotra, “A systematic review of machine learning techniques for software fault prediction,” Appl. Soft Computing, vol. 27, pp. 504-518, 2015.

[3] Matloob, F., Ghazal, T. M., Taleb, N., Aftab, S., Ahmad, M., Khan, M. A., … Soomro, T. R. (2021). Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review. IEEE Access, 9, 98754–98771.

[4] Song, Q., Guo, Y., & Shepperd, M. (2018). A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction. IEEE Transactions on Software Engineering, 1–1.

[5] A. B. Binkley and S. R. Schach, "Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures," *Proceedings of the 20th International Conference on Software Engineering*, Kyoto, Japan, 1998, pp. 452-455, doi: 10.1109/ICSE.1998.671604.

[6] S. R. Chidamber, D. P. Darcy and C. F. Kemerer, "Managerial use of metrics for object-oriented software: an exploratory analysis," in *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 629-639, Aug. 1998, doi: 10.1109/32.707698.

[7] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object-oriented design," in IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476-493, June 1994, doi: 10.1109/32.295895.

[8] Thota, M.K., Shajin, F.H., Rajesh, P. 2020. Survey on software defect prediction techniques. International Journal of Applied Science and Engineering, 17, 331–344.

[9] Rathore, Santosh S., and Sandeep Kumar. "An empirical study of ensemble techniques for software fault prediction." Applied Intelligence 51 (2021): 3615-3644.

[10] Goyal, Somya. "Handling class-imbalance with KNN (neighborhood) under-sampling for software defect prediction." Artificial Intelligence Review 55.3 (2022): 2023-2064.