International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

Class Imbalance Reduction and Centroid based Relevant Project Selection for Cross Project Defect Prediction

Kiran Kumar Bejjanki[1], Sai Priyanka Kanchanapally[2], Mahesh Kumar Thota[3]
[1]Associate Professor, Dept. of Information Technology, Kakatiya Institute of Technology & Science
Warangal, India
kiran_b_kumar@yahoo.com
[2]M.Tech Scholar, Dept. of Information Technology, Kakatiya Institute of Technology & Science
Warangal, India
M21DS003@kitsw.ac.in
[3]Assistant Professor, Dept. of Information Technology, Kakatiya Institute of Technology & Science
Warangal, India
thotamahi1262@gmail.com

Abstract— Cross-Project Defect Prediction (CPDP) is the process of predicting defects in a target project using information from other projects. This can assist developers in prioritizing their testing efforts and finding flaws. Transfer Learning (TL) has been frequently used at CPDP to improve prediction performance by reducing the disparity in data distribution between the source and target projects. Software Defect Prediction (SDP) is a common study topic in software engineering that plays a critical role in software quality assurance. To address the cross-project class imbalance problem, Centroid-based PF-SMOTE for Imbalanced data is used. In this paper, we used a Centroid-based PF-SMOTE to balance the datasets and Centroid based relevant data selection for Cross Project Defect Prediction. These methods use the mean of all attributes in a dataset and calculating the difference between mean of all datasets. For experimentation, the open source software defect datasets namely, AEEM, Re-Link, and NASA, are considered.

Keywords- Cross Project Defect Prediction; Class imbalance; PF-SMOTE; Software quality assurance**.**

I. INTRODUCTION

The popularity of software is rising dramatically in the present day. Customers won't be satisfied unless the software is flawless. Therefore, it is essential to check the software modules at the beginning of the process before proceeding to further stages. In order to determine whether there are any defects in the programme, it must be tested during the testing process. Using classification models created from past defect data from other projects, CPDP refers to the identification of potentially defect-prone modules in a project (i.e., target). There is typically a significant distribution mismatch between source and target projects since the programming languages, coding styles, and developer experience are different [1], [2]. Different issues can arise as a result of software flaws. Manual testing and code review are typical methods for locating software flaws. These techniques' fundamental flaw is that it takes a lot of time and effort to use them. By using automatic methods for Software Defect Prediction (SDP), one can cut costs while raising the calibre of software projects. Software Defect Prediction is a significant issue in the research of programming languages and software engineering. The objective is to accurately locate the defective code (in terms of the precision and recall) [3]. The majority of SDP techniques fall under the category of Within-Project Defect Prediction (WPDP), which creates a classification model using past data on defects to determine how prone to errors new software modules are within the same project [4], [5], [6], [7]. Although the aforementioned approaches have demonstrated the good CPDP prediction performance, there are a few limitations: Problems with parameter adjustment, cross-project class imbalance, and data selection [8]. In this paper, we used a Centroid-based PF-SMOTE to balance the datasets [31] and Centroid based relevant data selection for Cross Project Defect Prediction. The remaining sections are organized as follows: In Section II, the related work is reviewed. Section III details the recommended strategy. The experimental results are presented in Section IV. Finally, Section V presents the conclusions.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

II. RELATED WORK

The common software defect prediction methodology and the associated cross-project defect prediction techniques are briefly reviewed in this section.

A. Process for Predicting Software Defects

The goal of software defect prediction is to identify more frequently defective modules and make it possible for the organization's limited resources to be wisely distributed. The majority of the currently used defect prediction algorithms are based on machine learning methods [9], [10], which build prediction models using historical data on defects and then forecast the fault labels of unlabeled software modules.

B. Methods for Cross-Project Defect Prediction

Cross-project defect prediction uses the trained model with external project data to identify more likely defect-prone modules in a target project. CPDP has attracted the attention of an increasing number of academics in the field of software engineering [11], [12], [13]. A number of novel CPDP models have been created in recent years [14], [15], [16], [17]. Existing CPDP models can be split into two categories: training data selection based CPDP methods and transfer learning based CPDP methods.

C. Training data selection based CPDP methods

Finding the best training data for the target project data is the goal of training data selection-based CPDP algorithms [2]. The distribution gap between the source and target data can be minimized since the chosen source modules are comparable to the target project's modules.

D. Transfer learning based CPDP methods

Designing efficient algorithms with excellent generalization capabilities is the goal of transfer learning-based CPDP approaches [18], [19], [20], [21]. The majority of CPDP approaches concentrate on learning new metric representations for the source and target data in order to explicitly reduce the distribution difference between them and maybe bring the source's data distribution closer to the target's. Ma et al. [19] integrated transfer learning approach into CPDP and developed a transfer naive bayes method, which primarily transfers weight information from the source into the target, to reduce the distribution gap between the source and target data.

Other CPDP approaches [22], [23], [24], [25] focus on mixed-project defect prediction. However, these methods require a small number of tagged modules from the target project, which limits their application in the broader CPDP situation. Ryu et al. [26] applied a boosting-based model called value-cognitive boosting with support vector machine to the class imbalance problem in a cross-project defect prediction environment. Jing et al. [27] developed the SSTCA+ISDA method for dealing with the cross-project class imbalance problem by combining semi-supervised transfer component analysis with an improved subclass discriminant analysis algorithm.

E. Limitations of existing CPDP methods

Although the CPDP approaches detailed in the preceding section have made significant progress, there is still much opportunity for improvement. We discovered that the majority of existing CPDP approaches [2], [18], [28], [27], [20], [29] do not take into account the peculiarities of the defect prediction data (i.e., inconsistent distribution, class imbalance) and the hyper parametric optimization problem at the same time. Some of the training data selection-based CPDP approaches concentrate on picking relevant source data for the target at the module level. Instead of selecting similar projects with varying granularities at the project level, the other techniques chose similar projects with different granularities at the project level. However, these strategies simply took into account the inconsistency of the data distribution. The maximum mean discrepancy (MMD) [30] measure is used to directly calculate the similarity of source and target projects for unsupervised training data selection.

III. PROPOSED METHOD

In this section, we will go over the specifics of the proposed method, i.e., Class Imbalance Reduction and Centroid based relevant data selection for Cross Project Defect Prediction. To assess the efficacy of the examined techniques, AEEM, NASA and RELINK open source software defect datasets are taken into consideration for experimentation. The proposed architecture is given in Fig. 1.

A. Proposed Method

If the samples in the datasets are distributed unevenly i.e; majority and minority classes, occurs. To accomplish this, create synthetic samples. In this paper, we have considered the open source datasets such as AEEM, NASA and RELINK from different source projects. Preprocessing is a crucial step as we gather the datasets that must be altered in order for the algorithm to interpret them. The process is demonstrated below. The class label attribute of the source project is having values Y/N, T/F are changed to 1/0. As datasets are belonging to different projects, the software metrics may be considered different. So, to apply classification models the datasets should contain same type of metrics as attributes. We choose common attributes from the datasets for classification.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

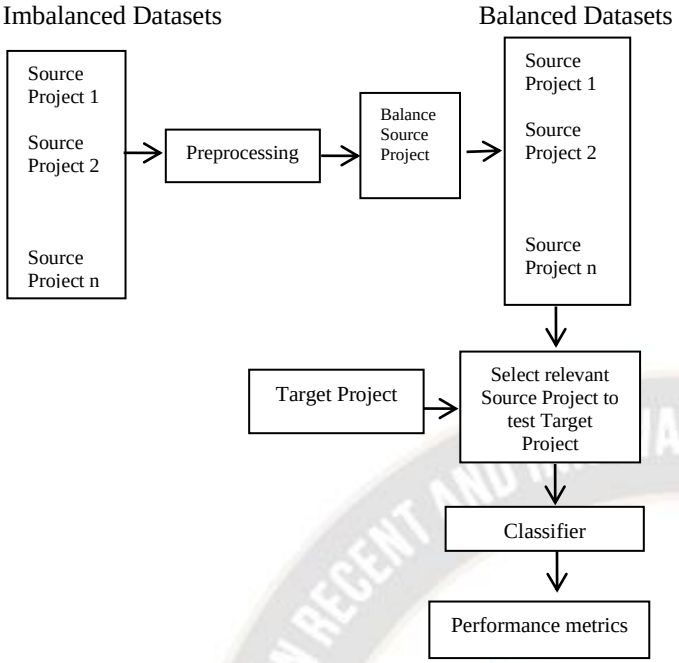

Fig. 1. Architecture of Proposed Method

B. Balance Source Projects

The datasets must be balanced during the training and testing phases. Section C describe the procedure used to balance the datasets.

C. Algorithm1: Balancing the datasets

Input: Minority class samples DSmin, Majority class samples DSmaj

Output: Synthetic samples Sg

1: DSmin_safe ← NULL, DSmin_bound← NULL

2: Calculate the Centroid of DSmin

3: Find k - nearest neighbours from Centroid

4: for each nearest neighbour ynn

4.1: if ynn ϵ DSmin then

4.2: DSmin_safe← DSmin_safe Uyi

4.3: else if ynn ϵ DSmaj then

4.4: DSmin_bound← DSmin_bound Uyi

end if

end for

5: Ssafe_g ← NULL

6: for all yi ϵ Ssafe_g, do

6.1: Find the local safe region LSR yi

6.2: Randomly select the sample ysafe ϵ LSR

6.3: generate synthetic samples Ssafe_g using

Ssafe_g = yi + random number between 0 and 1 × (ysafe - yi)

6.4: Ssafe_g ← Ssafe_g USsafe_g

end for

7: Sbound_g ← NULL

8: for all yi ϵ DSmin_bound, do

8.1: Find its nearest majority class sample ynn- ϵ DSmaj

8.2: generate synthetic samples Sboundg using

Ssafeg = yi + random number between 0 and 1 × (ysafe - yi)

8.3: Sbound_g ← Sbound_g U Sbound_g

end for

9: return Sg ← Ssafe_g U Sbound_g

D. Algorithm 2: Selection of relevant source project

Select the relevant source project to test the target project. The means of the datasets are first calculated in the suggested method, and the distance between the means of each dataset is then determined. The pseudo code for the suggested approach is given in section E.

E. Centroid based relevant project selection

Let $P_1, P_2, \ldots, P_n$ be the projects and $DS_1, DS_2, \ldots, DS_m$ be the datasets in each project P_i where $1 \leq i \leq n$.

Each dataset DS_j where $1 \leq j \leq m$ has $r_1, r_2, \ldots, r_p$ records with $a_1, a_2, \ldots, a_x$ attributes and $r_m a_n$ represents value of m^{th} *row* n^{th} *column.*

for each group of datasets belongs to same project :

Compute Centroid of each data set using $C=(\sum r_m a_n)/p$

Compute Euclidean distance between C *and* r_j *where* $1 \leq j \leq p$

Sort the distances in ascending order

Choose minimum distance dataset for training and testing

End for

F. Performance metrics

Accuracy, Precision, Recall, F-Measure, Specificity, Geometric Mean, Confusion matrix and AUC are some of the several performance metrics. The equations (1) to (7) show the metrics' formulae.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

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

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

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

F-Measure (FM) $$\frac{2*Precision*Recall}{Precision+Recall} \quad (4)$$

Specificity (SPEC) $$\frac{TN}{TN+FP} \quad (5)$$

Geometric Mean (GM) $$\sqrt{\frac{Recall*Specificity}{2}} \quad (6)$$

AUC $$\frac{TP}{TP+FN} \quad (7)$$

IV. EXPERIMENTAL RESULTS

This part aims to demonstrate the efficiency and dependability of the suggested solution, which is Class Imbalance Reduction (CIR) and Centroid based relevant project selection for Cross Project Defect Prediction. We considered the eight classifiers like Decision Tree, Logistic Regression, KNN, Naive Bayes, AdaBoost, Extra Tree, and SVM algorithms. Table I displays the imbalance ratio's of various datasets of the projects.

After applying Centroid based relevant project selection on the balanced datasets of each project shown in Table I, the minimum distances of mean values are shown in Table II to Table IV in bold face.

Table I: The Datasets Under Consideration

Project	Test dataset (TS)	Train dataset (TR)
AEEM	EQ	LUCENE
	JDT	MYLYN
	LUCENE	JDT
	MYLYN	PDE
	PDE	MYLYN
NASA	KC3	MC2
	MC2	KC3
	PC3	PC4
	PC4	PC3
RELINK	APACHE	SAFE
	SAFE	ZXING
	ZXING	SAFE

Table II. Distances Between Means of Datasets of AEEM Project

	EQ	JDT	LUCENE	MYLYN	PDE
EQ	0	639.30	**24.85**	43.90	28.71
JDT	639.30	0	681.0	**629.68**	658.25
LUCENE	**24.85**	629.68	0	56.48	33.16
MYLYN	43.90	681.0	56.48	0	**33.13**
PDE	28.71	658.25	33.16	**33.13**	0

Table III. Distances Between Means of Datasets of NASA Project

	KC3	MC2	PC3	PC4
KC3	0	**41244.17**	58410.26	49169.92
MC2	**41244.17**	0	91523.3	85953.74
PC3	58410.26	91523.35	0	**13867.99**
PC4	49169.9	85953.74	**13867.9**	0

Table IV. Distances Between Means of Datasets of RELINK Project

	APACHE	SAFE	ZXING
APACHE	0	**6.64**	6.68
SAFE	6.64	0	**1.71**
ZXING	6.64	**1.71**	0

The results shown in Table II to Table IV are used to select the source and target projects which are used to train and test the datasets, which are shown in Table V.

Table V. Training and Test Datasets

Project	Test dataset (TS)	Train dataset (TR)
AEEM	EQ	LUCENE
	JDT	MYLYN
	LUCENE	JDT
	MYLYN	PDE
	PDE	MYLYN
NASA	KC3	MC2
	MC2	KC3
	PC3	PC4
	PC4	PC3
RELINK	APACHE	SAFE
	SAFE	ZXING
	ZXING	SAFE

Upon choosing relevant project for training and test the datasets, various classification algorithms like Decision Tree, Logistic Regression, KNN, Naive Bayes, AdaBoost, Extra Tree, and SVM are used to calculate the performance measures using the equations 1 to 7. The results of our experiments are shown in table VI to table XII.

The experimentation results after applying AdaBoost algorithm on the datasets is shown in Table VI. It shows that, except some metrics most of the metrics are giving good performance for the combination of test and train datasets shown in table V.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

The experimentation results after applying Decision tree algorithm on the datasets is shown in Table VII. It shows that, except some metrics most of the metrics are giving good performance for the combination of test and train datasets shown in table V.

The experimentation results after applying Extra Tree algorithm on the datasets is shown in Table VIII. It shows that, except some metrics most of the metrics are giving good performance for the combination of test and train datasets shown in table V.

The experimentation results after applying KNN algorithm on the datasets is shown in Table IX. It shows that, except some metrics most of the metrics are giving good performance for the combination of test and train datasets shown in table V.

The experimentation results after applying Logistic Regression algorithm on the datasets is shown in Table X. It shows that, except some metrics most of the metrics are giving good performance for the combination of test and train datasets shown in table V.

The experimentation results after applying Naïve bayes on the datasets is shown in Table XI. It shows that, except some metrics most of the metrics are giving good performance for the combination of test and train datasets shown in table V.

The experimentation results after applying SVM algorithm on the datasets is shown in Table XII. It shows that, except some metrics most of the metrics are giving good performance for the combination of test and train datasets shown in Table V.

Table VI. Comparison of Performance Metrics Using AdaBoost Algorithm

TS	TR	ACC	PRE	REC	FM	SPEC	GM	AUC
EQ	**LUCENE**	**0.66**	**0.85**	0.17	0.28	**0.98**	**0.49**	**0.58**
	JDT	0.65	0.85	0.13	0.23	0.98	0.36	0.56
	MYLYN	0.66	0.71	**0.26**	**0.38**	0.93	0.49	0.6
	PDE	0.65	0.69	0.22	0.33	0.93	0.45	0.58
JDT	LUCENE	0.71	0.35	0.5	0.41	0.47	0.62	0.63
	EQ	0.41	0.25	**0.96**	0.4	0.27	0.51	0.61
	MYLYN	**0.93**	**0.6**	0.51	**0.55**	**0.91**	**0.69**	**0.71**
	PDE	0.78	0.47	0.57	0.52	0.83	0.69	0.70
LUCENE	EQ	0.91	0.48	**0.8**	0.33	0.98	**0.62**	0.64
	JDT	**0.91**	**0.48**	0.23	**0.34**	**0.98**	0.47	**0.65**
	MYLYN	0.89	0.27	0.09	0.14	0.97	0.3	0.53
	PDE	0.82	0.21	0.36	0.27	0.86	0.56	0.61
MYLYN	PDE	**0.88**	**0.37**	0.32	**0.27**	**0.97**	**0.55**	**0.59**
	EQ	0.3	0.13	**0.73**	0.22	0.23	0.41	0.48
	JDT	0.85	0.36	0.16	0.22	0.96	0.39	0.56
	LUCEN	0.63	0.17	0.45	0.25	0.66	0.54	0.56
	E							
PDE	EQ	0.5	0.17	0.68	0.27	0.47	0.57	0.58
	JDT	0.86	0.48	0.17	0.25	0.97	0.41	0.57
	LUCENE	0.84	0.35	0.21	0.26	0.94	0.44	0.57
	MYLYN	**0.87**	**0.50**	**0.74**	**0.38**	**0.97**	**0.58**	**0.63**
KC3	**MC2**	**0.87**	**0.29**	**0.22**	**0.24**	**0.89**	**0.42**	**0.55**
	PC3	0.81	0	0	0	1	0	0.5
	PC4	0.76	0.29	0.19	0.23	0.89	0.41	0.54
MC2	**KC3**	**0.66**	**0.25**	**0.07**	**0.11**	**0.89**	**0.25**	**0.48**
	PC3	0.65	0	0	0	1	0	0.5
	PC4	0.59	0.23	0.07	0.11	0.88	0.25	0.47
PC3	**PC4**	**0.88**	**0.26**	**0.14**	**0.18**	**0.94**	**0.36**	**0.54**
	KC3	0.87	0.23	0.02	0.04	0.93	0.14	0.51
	MC2	0.86	0.05	0.01	0.02	0.92	0.1	0.49
PC4	**PC3**	**0.86**	**0.7**	**0.07**	**0.11**	**0.98**	**0.26**	**0.52**
	KC3	0.85	0.5	0.01	0.02	0.9	0.1	0.5
	MC2	0.86	0.67	0.01	0.02	0.92	0.1	0.51
APACHE	**SAFE**	**0.7**	**0.69**	**0.73**	**0.71**	**0.76**	**0.69**	**0.7**
	ZXING	0.53	0.56	0.36	0.44	0.71	0.51	0.53
SAFE	**ZXING**	**0.68**	**0.64**	**0.41**	**0.5**	**0.85**	**0.59**	**0.63**
	APACHE	0.67	0.6	0.47	0.41	0.82	0.51	0.62
ZXING	**SAFE**	**0.76**	**0.48**	**0.36**	**0.39**	**0.9**	**0.53**	**0.57**
	APACHE	0.7	0.47	0.06	0.11	0.89	0.24	0.52

Table VII. Comparison of Performance Metrics Using Decision Tree Algorithm

TS	TR	ACC	PRE	REC	FM	SPEC	GM	AUC
EQ	**LUCENE**	**0.67**	**0.71**	0.23	**0.47**	**0.94**	**0.48**	**0.69**
	JDT	0.58	0.46	0.29	0.36	0.77	0.47	0.53
	MYLYN	0.67	0.68	**0.34**	0.45	0.89	0.55	0.62
	PDE	0.59	0.47	0.26	0.33	0.8	0.46	0.53
JDT	LUCENE	0.62	0.3	0.64	0.41	0.62	0.63	0.63
	EQ	0.47	0.25	**0.81**	0.38	0.38	0.55	0.59
	MYLYN	**0.67**	**0.37**	0.62	**0.43**	**0.66**	**0.69**	**0.69**
	PDE	0.54	0.28	0.74	0.41	0.49	0.6	0.62
LUCENE	EQ	0.5	0.12	0.7	0.2	0.48	0.58	0.59
	JDT	**0.85**	**0.24**	**0.39**	**0.29**	**0.9**	**0.59**	**0.59**
	MYLYN	0.84	0.24	0.34	0.28	0.89	0.55	0.62
	PDE	0.71	0.14	0.39	0.21	0.75	0.54	0.57
MYLYN	**PDE**	**0.86**	**0.19**	**0.59**	**0.37**	**0.89**	**0.58**	**0.59**
	EQ	0.4	0.11	0.51	0.18	0.39	0.45	0.45
	JDT	0.62	0.14	0.37	0.2	0.66	0.49	0.51
	LUCENE	0.81	0.13	0.32	0.31	0.88	0.53	0.58

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

PDE	EQ	0.39	0.13	**0.61**	0.21	0.36	0.47	0.49
	JDT	0.78	0.27	0.32	0.29	0.86	0.52	0.59
	LUCENE	0.8	0.31	0.37	0.34	0.87	0.57	0.62
	MYLYN	**0.85**	**0.32**	0.38	**0.39**	**0.89**	**0.58**	**0.69**
KC3	**MC2**	**0.85**	**0.28**	**0.53**	**0.3**	**0.98**	**0.54**	**0.61**
	PC3	0.76	0.25	0.14	0.18	0.91	0.36	0.52
	PC4	0.84	0.27	0.19	0.3	0.98	0.43	0.59
MC2	**KC3**	**0.64**	**0.35**	**0.39**	**0.47**	**0.92**	**0.59**	**0.57**
	PC3	0.62	0.44	0.36	0.4	0.75	0.52	0.56
	PC4	0.6	0.31	0.11	0.16	0.86	0.31	0.49
PC3	**PC4**	**0.82**	**0.24**	**0.21**	**0.22**	**0.91**	**0.44**	**0.56**
	KC3	0.35	0.1	0.5	0.17	0.33	0.41	0.41
	MC2	0.76	0.11	0.13	0.12	0.85	0.33	0.49
PC4	**PC3**	**0.82**	**0.21**	**0.16**	**0.19**	**0.91**	**0.38**	**0.54**
	KC3	0.32	0.11	0.12	0.19	0.28	0.42	0.45
	MC2	0.72	0.05	0.08	0.06	0.8	0.25	0.44
APACHE	**SAFE**	**0.64**	**0.61**	**0.78**	**0.68**	**0.5**	**0.62**	**0.64**
	ZXING	0.56	0.55	0.72	0.62	0.3	0.53	0.55
SAFE	**ZXING**	**0.77**	**0.76**	**0.68**	**0.69**	**0.9**	**0.75**	**0.75**
	APACHE	0.68	0.75	0.27	0.4	0.9	0.5	0.61
ZXING	**SAFE**	**0.7**	**0.49**	**0.35**	**0.37**	**0.7**	**0.52**	**0.56**
	APACHE	0.69	0.44	0.18	0.26	0.6	0.4	0.54

Table VIII. Comparison of Performance Metrics Using Extra Tree Algorithm

TS	TR	ACC	PRE	REC	FM	SPEC	GM	AUC
EQ	**LUCENE**	**0.64**	**0.7**	0.15	**0.35**	**0.96**	**0.48**	**0.55**
	JDT	0.58	0.44	0.23	0.3	0.81	0.43	0.52
	MYLYN	0.58	0.44	0.19	0.27	0.84	0.4	0.51
	PDE	0.56	0.42	**0.29**	0.34	0.73	0.46	0.51
JDT	LUCENE	0.67	0.23	0.45	0.44	0.85	0.62	0.65
	EQ	0.43	0.22	**0.7**	0.33	0.36	0.5	0.53
	MYLYN	**0.67**	**0.28**	0.38	**0.52**	**0.87**	**0.63**	**0.66**
	PDE	0.58	0.25	0.52	0.34	0.6	0.56	0.56
LUCENE	EQ	0.55	0.09	0.42	0.15	0.56	0.48	0.49
	JDT	**0.86**	**0.24**	**0.46**	**0.29**	**0.88**	**0.6**	**0.62**
	MYLYN	0.85	0.22	0.23	0.22	0.82	0.46	0.58
	PDE	0.8	0.21	0.41	0.28	0.84	0.59	0.62
MYLYN	**PDE**	**0.77**	**0.29**	0.34	**0.34**	**0.88**	**0.61**	**0.66**
	EQ	0.3	0.11	0.63	0.19	0.25	0.4	0.44
	JDT	0.74	0.18	0.29	0.22	0.81	0.48	0.55
	LUCENE	0.76	0.26	**0.45**	0.33	0.81	0.6	0.63
PDE	EQ	0.58	0.17	**0.54**	0.26	0.59	0.56	0.56
	JDT	0.8	0.25	0.22	0.23	0.89	0.44	0.56
	LUCENE	0.8	0.24	0.2	0.22	0.9	0.42	0.55
	MYLYN	**0.82**	**0.3**	0.4	**0.27**	**0.9**	**0.57**	**0.59**
KC3	**MC2**	**0.86**	**0.29**	**0.53**	**0.36**	**0.77**	**0.57**	**0.59**
	PC3	0.81	0	0	0	1	0	0.5
	PC4	0.7	0.28	0.42	0.34	0.76	0.56	0.59
MC2	**KC3**	**0.77**	**0.53**	**0.66**	**0.52**	**0.82**	**0.59**	**0.59**
	PC3	0.66	0.52	0.39	0.45	0.8	0.56	0.59
	PC4	0.47	0.31	0.41	0.35	0.51	0.46	0.46
PC3	**PC4**	**0.78**	**0.21**	0.27	**0.34**	**0.85**	**0.69**	**0.71**
	KC3	0.59	0.21	**0.82**	0.33	0.56	0.68	0.69
	MC2	0.36	0.13	0.74	0.22	0.31	0.48	0.52
PC4	**PC3**	**0.83**	**0.24**	**0.27**	**0.2**	**0.93**	**0.44**	**0.55**
	KC3	0.81	0.03	0.02	0.02	0.92	0.14	0.47
	MC2	0.76	0.16	0.22	0.19	0.84	0.43	0.53
APACHE	**SAFE**	**0.68**	**0.67**	**0.71**	**0.69**	**0.65**	**0.68**	**0.68**
	ZXING	0.62	0.62	0.61	0.61	0.62	0.61	0.62
SAFE	**ZXING**	**0.73**	**0.77**	**0.64**	**0.65**	**0.89**	**0.71**	**0.72**
	APACHE	0.73	0.71	0.55	0.62	0.85	0.68	0.7
ZXING	**SAFE**	**0.69**	**0.43**	**0.44**	**0.42**	**0.87**	**0.57**	**0.58**
	APACHE	0.68	0.42	0.23	0.3	0.87	0.45	0.55

Table IX. Comparison of Performance Metrics Using KNN Algorithm

TS	TR	ACC	PRE	REC	FM	SPEC	GM	AUC
EQ	**LUCENE**	**0.66**	**0.91**	**0.09**	**0.25**	**0.99**	0.28	**0.59**
	JDT	0.65	0.77	0.19	0.3	0.96	**0.43**	0.58
	MYLYN	0.63	0.69	0.14	0.23	0.96	0.37	0.55
	PDE	0.63	0.8	0.09	0.16	0.98	0.3	0.54
JDT	LUCENE	0.78	0.38	0.12	0.18	0.75	0.34	0.54
	EQ	0.45	0.24	0.8	0.37	0.35	0.53	0.58
	MYLYN	**0.89**	0.28	**0.82**	**0.39**	**0.79**	**0.55**	**0.59**
	PDE	0.81	**0.61**	0.2	0.3	0.77	0.44	0.58
LUCENE	EQ	0.63	0.14	0.06	0.22	0.63	0.59	0.6
	JDT	**0.87**	**0.39**	**0.12**	**0.25**	**0.98**	**0.64**	**0.61**
	MYLYN	0.87	0.11	0.06	0.08	0.95	0.24	0.5
	PDE	0.9	0.29	0.08	0.13	0.98	0.28	0.53
MYL	**PDE**	**0.86**	**0.48**	0.12	**0.28**	**0.97**	**0.64**	**0.59**

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

YN	EQ	0.64	0.18	**0.5**	0.26	0.66	0.57	0.58
	JDT	0.86	0.43	0.13	0.2	0.97	0.36	0.55
	LUCENE	0.85	0.27	0.07	0.11	0.97	0.26	0.52
PDE	EQ	0.54	0.17	0.07	0.26	0.54	**0.55**	0.55
	JDT	0.85	0.38	0.14	0.2	0.96	0.37	0.55
	LUCENE	0.85	0.42	0.11	0.17	0.98	0.33	0.54
	MYLYN	**0.86**	**0.44**	**0.15**	**0.27**	**0.98**	0.38	**0.55**
KC3	**MC2**	**0.88**	**0.38**	**0.31**	**0.34**	0.98	**0.53**	**0.6**
	PC3	0.8	0.38	0.08	0.13	0.97	0.28	0.53
	PC4	0.81	0	0	0	**1**	0	0.5
MC2	**KC3**	**0.68**	**0.67**	**0.18**	**0.28**	0.95	**0.41**	**0.57**
	PC3	0.66	0.58	0.16	0.25	0.94	0.39	0.55
	PC4	0.65	0	0	0	**1**	0	0.5
PC3	**PC4**	**0.88**	**0.21**	**0.08**	**0.12**	0.96	**0.28**	**0.52**
	KC3	0.87	0	0	0	**1**	0	0.5
	MC2	0.88	0.5	0.01	0.02	**1**	0.1	0.5
PC4	**PC3**	**0.89**	**0.23**	**0.05**	**0.08**	0.98	**0.22**	**0.51**
	KC3	0.88	1	0.01	0.02	**1**	0.1	0.5
	MC2	0.88	1	0.01	0.02	**1**	0.1	0.5
APACHE	**SAFE**	**0.72**	**0.7**	**0.8**	**0.75**	**0.65**	**0.72**	**0.72**
	ZXING	0.64	0.64	0.68	0.66	0.6	0.64	0.64
SAFE	**ZXING**	**0.72**	**0.75**	**0.41**	**0.56**	**0.96**	**0.66**	**0.69**
	APACHE	0.71	0.75	0.41	0.53	0.91	0.61	0.66
ZXING	**SAFE**	**0.69**	**0.46**	**0.18**	**0.27**	**0.92**	**0.4**	**0.55**
	APACHE	0.69	0.45	0.18	0.26	0.91	0.4	0.54

Table X. Comparison of Performance Metrics Using Logistic Regression Algorithm

TS	TR	ACC	PRE	REC	FM	SPEC	GM	AUC
EQ	**LUCENE**	**0.69**	**0.8**	**0.38**	**0.47**	**0.95**	**0.56**	**0.63**
	JDT	0.63	0.62	0.18	0.28	0.93	0.41	0.55
	MYLYN	0.67	0.73	0.27	0.39	0.93	0.5	0.6
	PDE	0.69	0.77	0.32	0.45	0.94	0.55	0.63
JDT	LUCENE	0.8	0.51	0.45	0.48	0.89	0.63	0.67
	EQ	0.4	0.24	0.87	0.38	0.27	0.48	0.57
	MYLYN	**0.88**	**0.59**	**0.87**	**0.52**	0.53	**0.67**	**0.69**
	PDE	0.81	0.55	0.48	0.51	**0.9**	0.66	0.69
LUCENE	EQ	0.73	0.17	**0.48**	0.25	0.75	0.6	0.62
	JDT	**0.9**	**0.36**	0.19	**0.27**	**0.97**	**0.62**	**0.67**
	MYLYN	0.89	0.32	0.17	0.22	0.96	0.4	0.57
	PDE	0.87	0.32	0.33	0.32	0.93	0.55	0.63

MYLYN	**PDE**	**0.84**	**0.38**	**0.41**	**0.37**	**0.92**	**0.59**	**0.66**
	EQ	0.68	0.13	0.25	0.17	0.74	0.43	0.5
	JDT	0.83	0.36	0.37	0.36	0.9	0.58	0.63
	LUCENE	0.68	0.18	0.4	0.25	0.72	0.54	0.56
PDE	EQ	0.57	0.19	**0.66**	0.3	0.55	0.6	0.6
	JDT	0.85	0.44	0.24	0.31	0.95	0.48	0.6
	LUCENE	0.8	0.31	0.33	0.32	0.88	0.54	0.61
	MYLYN	**0.86**	**0.45**	0.34	**0.34**	**0.95**	**0.65**	**0.62**
KC3	**MC2**	**0.8**	**0.44**	0.22	**0.39**	**0.94**	**0.45**	**0.58**
	PC3	0.19	0.19	**1**	0.32	0.2	0.2	0.5
	PC4	0.19	0.19	**1**	0.32	0.3	0.31	0.5
MC2	**KC3**	**0.66**	**0.6**	0.07	**0.53**	**0.98**	**0.26**	**0.52**
	PC3	0.35	0.35	**1**	0.52	0	0	0.5
	PC4	0.36	0.35	**1**	0.52	0.01	0.1	0.51
PC3	**PC4**	**0.89**	**0.27**	**0.19**	**0.22**	0.93	**0.42**	**0.56**
	KC3	0.88	0	0	0	**1**	0	0.5
	MC2	0.88	0	0	0	**1**	0	0.5
PC4	**PC3**	**0.86**	**0.26**	**0.06**	**0.1**	**0.98**	**0.24**	**0.52**
	KC3	0.84	0	0	0	0.95	0	0.48
	MC2	0.8	0	0	0	0.92	0	0.46
APACHE	**SAFE**	**0.67**	**0.67**	0.47	**0.72**	**0.64**	**0.65**	**0.65**
	ZXING	0.65	0.62	**0.81**	0.7	0.49	0.63	0.65
SAFE	**ZXING**	**0.71**	**0.62**	0.68	**0.65**	**0.74**	**0.71**	**0.71**
	APACHE	0.39	0.39	**1**	0.56	0	0	0.5
ZXING	**SAFE**	**0.68**	**0.37**	0.11	**0.47**	**0.92**	**0.32**	**0.52**
	APACHE	0.3	0.3	**1**	0.46	0	0	0.5

Table XI. Comparison of Performance Metrics Using Naïve Bayes Algorithm

TS	TR	ACC	PRE	REC	FM	SPEC	GM	AUC
EQ	**LUCENE**	**0.72**	**0.84**	**0.46**	**0.58**	**0.96**	**0.68**	**0.67**
	JDT	0.69	0.83	0.26	0.4	0.96	0.5	0.61
	MYLYN	0.71	0.76	0.4	0.52	0.92	0.61	0.66
	PDE	0.71	0.79	0.36	0.49	0.94	0.58	0.65
JDT	LUCENE	0.51	0.29	**0.91**	0.44	0.41	0.61	0.66
	EQ	0.41	0.25	0.94	0.39	0.28	0.51	0.61
	MYLYN	**0.54**	**0.32**	0.85	**0.46**	**0.46**	**0.63**	**0.68**
	PDE	0.54	0.3	0.89	0.45	0.45	0.63	0.67
LUCENE	EQ	0.77	0.22	**0.59**	0.32	0.79	0.68	0.69
	JDT	**0.89**	**0.39**	0.27	**0.38**	**0.96**	**0.69**	**0.69**
	MYLY	0.87	0.33	0.42	0.37	0.91	0.62	0.67

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

	N							
	PDE	0.86	0.32	0.45	0.37	0.9	0.64	0.68
MYLYN	**PDE**	**0.77**	**0.33**	0.67	**0.45**	**0.82**	**0.64**	**0.67**
	EQ	0.3	0.13	0.79	0.22	0.23	0.43	0.51
	JDT	0.74	0.24	0.44	0.31	0.78	0.59	0.61
	LUCENE	0.51	0.29	**0.91**	0.44	0.41	0.61	0.66
PDE	EQ	0.63	0.21	**0.61**	0.31	0.64	0.62	0.62
	JDT	0.84	0.38	0.29	0.33	0.92	0.52	0.61
	LUCENE	0.77	0.29	0.47	0.36	0.82	0.62	0.64
	MYLYN	**0.85**	**0.39**	0.43	**0.37**	**0.96**	**0.62**	**0.64**
KC3	**MC2**	**0.8**	**0.44**	0.19	**0.37**	**0.99**	**0.42**	**0.57**
	PC3	0.8	0	0	0	0.99	0	0.49
	PC4	0.2	0.19	**1**	0.32	0.02	0.14	0.51
MC2	**KC3**	**0.73**	**0.66**	0.48	**0.56**	**0.89**	**0.64**	**0.67**
	PC3	0.58	0	0	0	0.89	0	0.44
	PC4	0.42	0.38	**1**	0.55	0.11	0.33	0.56
PC3	**PC4**	**0.89**	**0.25**	0.12	**0.26**	**1**	**0.34**	**0.53**
	KC3	0.14	0.13	**1**	0.23	0.02	0.14	0.51
	MC2	0.88	0	0	0	1	0	0.5
PC4	**PC3**	0.25	**0.14**	**0.97**	**0.24**	0.15	**0.48**	**0.58**
	KC3	0.28	0.14	0.96	0.24	0.18	0.42	0.57
	MC2	**0.87**	0	0	0	**1**	0	0.5
APACHE	**SAFE**	**0.72**	**0.69**	**0.87**	**0.75**	**0.62**	**0.71**	**0.72**
	ZXING	0.66	0.62	0.86	0.72	0.46	0.63	0.66
SAFE	**ZXING**	**0.7**	**0.6**	**0.68**	**0.64**	0.71	**0.69**	**0.69**
	APACHE	0.61	0	0	0	**1**	0	0.5
ZXING	**SAFE**	**0.7**	**0.49**	**0.19**	**0.27**	0.91	**0.42**	**0.55**
	APACHE	0.7	0	0	0	**1**	0	0.5

Table XII. Comparison of Performance Metrics Using SVM Algorithm

TS	TR	ACC	PRE	REC	FM	SPEC	GM	AUC
EQ	**LUCENE**	**0.65**	0	0	0	**1**	**0.5**	**0.55**
	JDT	0.64	**1**	**0.09**	**0.17**	1	0.3	0.54
	MYLYN	0.62	0.86	0.05	0.09	0.99	0.22	0.52
	PDE	0.61	1	0.02	0.04	1	0.14	0.51
JDT	LUCENE	0.8	1	0.01	0.02	1	0.1	0.5
	EQ	0.56	0.29	0.08	0.12	0.51	0.3	**0.64**
	MYLYN	**0.81**	**1**	**0.09**	**0.16**	**1**	**0.3**	0.54
	PDE	0.79	0.5	0	0	1	0	0.5
LUC	EQ	0.81	0.22	**0.42**	0.29	0.85	**0.6**	0.63
ENE	**JDT**	**0.92**	**0.3**	0.05	**0.39**	**0.99**	0.22	**0.64**
	MYLYN	0.9	0	0	0	0.99	0	0.5
	PDE	0.91	0	0	0	1	0	0.5
MYLYN	**PDE**	**0.87**	0	**0.4**	**0.4**	**1**	0	**0.65**
	EQ	0.81	0.3	0.36	0.33	0.87	**0.56**	0.62
	JDT	0.87	**0.53**	0.03	0.06	1	0.17	0.51
	LUCENE	0.87	0.5	0	0	1	0	0.5
PDE	EQ	0.73	0.25	0.48	0.33	0.77	0.61	0.62
	JDT	0.86	0.62	0.02	0.04	**1**	0.14	0.51
	LUCENE	0.86	0	0	0	**1**	0	0.5
	MYLYN	**0.86**	**0.69**	**0.52**	**0.44**	0.99	**0.14**	**0.51**
KC3	**MC2**	**0.81**	**0.47**	**0.19**	**0.27**	0.95	**0.42**	**0.57**
	PC3	0.81	0	0	0	**1**	0	0.5
	PC4	0.81	0	0	0	**1**	0	0.5
MC2	**KC3**	**0.65**	**0**	**0**	**0**	**1**	**0**	**0.5**
	PC3	0.65	0	0	0	1	0	0.5
	PC4	0.65	0	0	0	1	0	0.5
PC3	**PC4**	**0.89**	**0**	**0**	**0**	**1**	**0**	**0.5**
	KC3	0.88	0	0	0	1	0	0.5
	MC2	0.85	0.12	0.04	0.06	0.96	0.2	0.5
PC4	**PC3**	**0.88**	**0**	**0**	**0**	**1**	**0**	**0.5**
	KC3	0.88	0	0	0	1	0	0.5
	MC2	0.86	0.2	0.04	0.07	0.98	0.2	0.51
APACHE	**SAFE**	**0.73**	**0.69**	**0.84**	**0.76**	0.62	**0.72**	**0.73**
	ZXING	0.48	0.29	0.02	0.04	**0.95**	0.14	0.48
SAFE	**ZXING**	**0.66**	**0.8**	**0.18**	0.29	**0.97**	**0.42**	**0.58**
	APACHE	0.39	0.39	1	**0.56**	0	0	0.5
ZXING	**SAFE**	**0.72**	**0.59**	**0.14**	0.23	**0.96**	**0.37**	**0.55**
	APACHE	0.3	0.3	1	**0.46**	0	0	0.5

V. CONCLUSION

In this paper, we applied the algorithm to balance imbalanced datasets and Centroid based relevant project selection algorithm for Cross Project Defect Prediction. The AEEM, Re-Link, and NASA open-source software defect datasets are taken into consideration for experimentation. The experimentation results after applying various algorithms on the datasets related to different projects shows that, except some metrics, most of the metrics are giving good performance for the combination of test and train datasets.

In future, we would like to consider more datasets from various open-source projects for experimentation.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

REFERENCES

[1] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," in FSE/ESEC'09. ACM, 2009, pp. 91–100.

[2] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," Empirical Software Engineering, vol. 14, no. 5, pp. 540–578, 2009.

[3] A Survey on Software Defect Prediction Using Deep Learning Elena N. Akimova 1,2,∗ , Alexander Yu. Bersenev 1,2, Artem A. Deikov 1,2, Konstantin S. Kobylkin 1,2 , Anton V. Konygin 1 , Ilya P. Mezentsev 1,2 and Vladimir E. Misilov 1,2.

[4] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," IEEE Transactions on Software Engineering, vol. 34, no. 4, pp. 485–496, 2008.

[5] X.-Y. Jing, S. Ying, Z.-W. Zhang, S.-S. Wu, and J. Liu, "Dictionary learning based software defect prediction," in ICSE'14. ACM, 2014, pp. 414–423.

[6] T. Wang, Z. Zhang, X.-Y. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," Automated Software Engineering, vol. 23, no. 4, pp. 569–590, 2016.

[7] Z. Xu, J. Liu, X. Luo, Z. Yang, Y. Zhang, P. Yuan, Y. Tang, and T. Zhang, "Software defect prediction based on kernel PCA and weighted extreme learning machine," Information and Software Technology, vol. 106, pp. 182–200, 2019.

[8] DSSDPP: Data Selection and Sampling based Domain Programming Predictor for Cross-project Defect Prediction Zhiqiang Li, Hongyu Zhang, Xiao-Yuan Jing, Juanying Xie, Min Guo, Jie Ren.

[9] Z. Li, X.-Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," IET Software, vol. 12, no. 3, pp. 161–175, 2018.

[10] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," IEEE Transactions on Software Engineering, vol. 40, no. 6, pp. 603–616, 2014.

[11] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y. Li, Y. Zhao, J. Qian, and B. Xu, "How far we have progressed in the journey? An examination of cross-project defect prediction," ACM Transactions on Software Engineering and Methodology, vol. 27, no. 1, pp. 1–51, 2018.

[12] S. Herbold, A. Trautsch, and J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches," IEEE Transactions on Software Engineering, vol. 44, no. 9, pp. 811–833, 2018.

[13] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," IEEE Transactions on Software Engineering, vol. 45, no. 2, pp. 111– 147, 2019.

[14] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter language reuse," in PROMISE'08, 2008, pp. 19–24

[15] A. E. Camargo Cruz and K. Ochimizu, "Towards logistic regression models for predicting fault-prone code across software projects," in ESEM'09, 2009, pp. 460–463.

[16] C. Ni, W. S. Liu, X. Chen, Q. Gu, D. X. Chen, and Q. G. Huang, "A cluster based feature selection method for cross-project software defect prediction," Journal of Computer Science and Technology, vol. 32, no. 6, pp. 1090–1107, 2017.

[17] Y. Zhang, L. O. David, X. Xia, and J. Sun, "Combined classifier for cross-project defect prediction: an extended empirical study," Frontiers of Computer Science, vol. 12, no. 2, pp. 280–296, 2018.

[18] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in ICSE'13. IEEE, 2013, pp. 382–391.

[19] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross company software defect prediction," Information and Software Technology, vol. 54, no. 3, pp. 248–256, 2012.

[20] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," Information and Software Technology, vol. 107, pp. 125–136, 2019.

[21] Z. Li, J. Niu, X.-Y. Jing, W. Yu, and C. Qi, "Cross-project defect prediction via landmark selection-based kernelized discriminant subspace alignment," IEEE Transactions on Reliability, vol. 70, no. 3, pp. 996–1013, 2021.

[22] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "Hydra: massively compositional model for cross-project defect prediction," IEEE Transactions on Software Engineering, vol. 42, no. 10, pp. 977– 998, 2016

[23] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," Information and Software Technology, vol. 62, pp. 67–77, 2015.

[24] D. Ryu, J.-I. Jang, and J. Baik, "A transfer cost-sensitive boosting approach for cross-project defect prediction," Software Quality Journal, vol. 25, no. 1, pp. 235–272, 2017.

[25] L. Gong, S. Jiang, and L. Jiang, "An improved transfer adaptive boosting approach for mixed-project defect prediction," Journal of Software: Evolution and Process, vol. 31, no. 10, pp. 1–28, 2019.

[26] D. Ryu, O. Choi, and J. Baik, "Value-cognitive boosting with a support vector machine for cross-project defect prediction," Empirical Software Engineering, vol. 21, no. 1, pp. 43–71, 2016.

[27] Shankarpure, M. R. ., & Patil, D. D. . (2023). A Comprehensive Survey on Methods and Techniques for Automated Fruit Plucking. International Journal of Intelligent Systems and Applications in Engineering, 11(1), 156–168. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2454.

[28] X.-Y. Jing, F. Wu, X. Dong, and B. Xu, "An improved sda based defect prediction framework for both within-project and cross project class-imbalance problems," IEEE Transactions on Software Engineering, vol. 43, no. 4, pp. 321–338, 2017.

[29] D. Ryu, J. Jang, and J. Baik, "A hybrid instance selection using nearest-neighbor for cross-project defect prediction," Journal of Computer Science and Technology, vol. 30, no. 5, pp. 969–980, 2015.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6933
***Article Received**: 29 March 2023 **Revised**: 15 May 2023 **Accepted**: 30 May 2023*

[30] Z. Xu, S. Pang, T. Zhang, X. Luo, J. Liu, Y. Tang, X. Yu, and L. Xue, "Cross project defect prediction via balanced distribution adaptation based transfer learning," Journal of Computer Science and Technology, vol. 34, no. 5, pp. 1039–1062, 2019.

[31] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schlkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in NIPS'07. MIT Press, 2007, pp. 513–520.

[32] Kiran Kumar Bejjanki, Sai Priyanka Kanchanapally, "Centroid-based PF-SMOTE for Imbalanced data," International Conference on Mathematical Sciences and Emerging Applications in Technology (ICMSEAT-2022) (In collaboration with APTSMS), September 9-11, 2022

Author's Profile:

B. Kiran Kumar received the Ph. D in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad in 2021. He is currently working as Associate Professor in the Department of Information Technology, Kakatiya Institute of Technology & Science, Warangal, Telangana, India. His research interest includes Data Mining, Machine learning, Software engineering.

K. Sai Priyanka pursuing M. Tech in Data Science in the Department of Information Technology, Kakatiya Institute of Technology & Science, Warangal, Telangana, India. Her research interest includes Data Mining, Machine learning, Software engineering.

Mahesh Kumar Thota pursuing Ph.D in CSE from University Visesvaraya College of Engineering, Bangalore University, Bangalore. He is currently working as Assistant Professor in the Department of Information Technology, Kakatiya Institute of Technology & Science, Warangal, Telangana, India. His research interest includes Internet of Things, Machine learning, Software engineering.