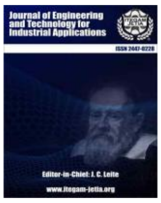
Journal of Engineering and Technology for Industrial Applications

ISSN ONLINE: 2447-0228

ITEGAM-JETIA

Manaus, v.10n.48, p. 92-102. July/August., 2024.
DOI: https://doi.org/10.5935/jetia. v10i48.991



RESEARCH ARTICLEOPEN ACCESS

# SOFTWARE DEFECT PREDICTION USING GLOBAL AND LOCAL MODELS

*Vikas Suhag[1], Sanjay Kumar Dubey[2] and Bhupendra Kumar Sharma[3]

[1,2] Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, India.
[3] Northern India Textile Research Association, Ghaziabad, Uttar Pradesh, India.

[1] http://orcid.org/0000-0002-6341-6375, [2] http://orcid.org/0000-0003-3808-6623, [3] http://orcid.org/0009-0003-1577-3647

Email: *vikasuhag@hotmail.com[1] , skdubey1@amity.edu[2] , drbkjpr@ymail.com[3]

**ARTICLE INFO**

***Article History***
Received: December 08th, 2024
Reviseed: July 08th, 2024
Accepted: July 08th, 2024
Published: July 18th, 2024

***Keywords:***
Local models,
global models,
defect prediction,
machine learning,
feature selection.

**ABSTRACT**

Despite intense investigation in the area of software defect prediction, there are some critical regions that still need attention. Heterogeneity of data is one of these areas that seek attention. Local models have gained focus in resolving the problem of heterogeneity. Limited studies have proven local models to be better than global models, so there is contradiction among researcher. Various researchers also considered feature selection as a method to mitigate the affect of heterogeneity. Our study presents a hybrid feature selection strategy with global and local (GL) models of software defect prediction (SDP). The proposed Hybrid Feature Selection Strategy (HFSS) has additionally improved the predicting power of GL models. Empirical results showcase that local models have preferential results than global models. Our study compared proposed approach with baselines techniques from literature on three PROMISE projects and traditional global models. Our proposed approach achieved better results in terms of accuracy, precision, recall and f-measure.

## I. INTRODUCTION

With growing complexity of the software, it is generally hard to develop software that is free from anomalies or defects. Errors or bugs might creep in during development due to logical or typographical errors. These errors must be detected timely for the reduction in maintenance cost of the software and ensure timely delivery of the project. For years, the companies have been deploying various testing techniques to identify bugs in the software[1]. But as the project size increases, testing becomes complex, costly, time consuming and finding defects becomes tedious.

Shippey [2] utilized the Abstract Syntax Tree n-grams to identify fault inducing code features. Zhou [3] developed deep forest models using the cascade strategy of ensemble learning (EL) and deep learning for effective SDP. Jayanthi & Florence [4] presented solution for improving performance of software defect prediction (SDP) by combining feature reduction and classification using artificial neural network. D. Rodriguez et. al. [5] proposed Zero Inflated Poisson (ZIP) models for SDP using R packages. Ostrand et. al. [6] and Borandag et. al. [7] proposed SDP using negative binomial regression and majority vote feature selection based Naive Bayes (NB), k-nearest Neighbour (KNN), J48 models respectively. Therefore, recently researchers started working on software defect prediction.

SDP is one kind of software quality assurance techniques, which aims to detect proneness to defect by learning from defect data [8]. SDP helps in optimal allocation of resources for development and maintenance. Various SDP techniques have arisen over the time viz. Cross Project Defect Prediction (CPDP) [9], Within Project Defect Prediction (WPDP) [10], and techniques based on local models [11], global models [11], few shot learning [12], transfer learning [13] etc.

Most of defect prediction techniques use machine learning models to predict bugs. Machine learning models uses labelled source code metrics data to train and test the model. Herbold et.al. [14] used several combinations of complexity metrics such as lines of code (LOC), average method complexity (AMC), McCabe's cyclomatic complexity (MCC), maximum and average cyclomatic complexity among methods, hallstead metrics. Nam et.al. [15] used LOC [16], MCC, halstead and various other metrics are used by rearchers are number of additions[17], number of deletions[17]. Analysing such source code metrics can give useful insights about the code, so as to predict future defects. Thus, SDP helps us recognize bugs in source code by analyzing the source code metrics data.

Most of the studies on SDP have used the machine learning models on the entire dataset of project, where data values are not evenly distributed, causing the problem of heterogeneity within the dataset. Heterogeneity is mainly caused by the different measurements standards and data differences between projects. But dataset contain some internal regions of data, which are homogeneous in nature. These homogeneous units of data within projects can help resolve heterogeneity issues. So homogeneous regions are created using clusters and then the modelling techniques are applied on those regions. The resulting models are termed as local models. However, in global models, the prediction models are applied over the complete dataset as a single unit as shown in Fig. 1 [18].

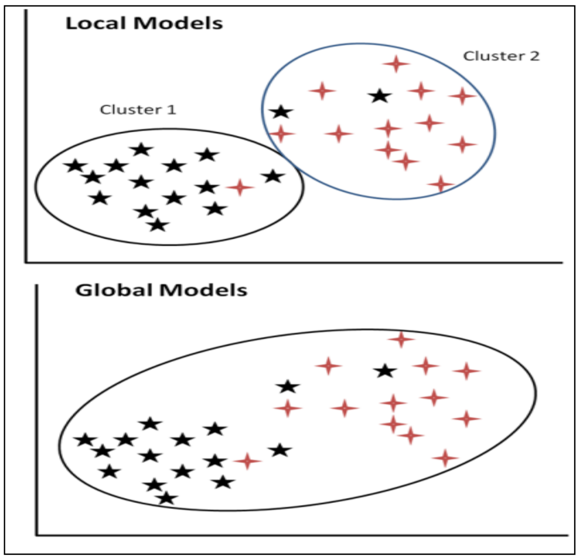


Figure 1: GL models
Source: Authors, (2024).

**Research questions:** In our study, we provided a hybrid feature selection strategy (HFSS) to find the most important feature set from dataset, local models to mitigate the effect of heterogeneity of data. Just in Time (JIT) GL models integrated with end to end software defect prediction framework for real-time defect prediction using commit level data. So we set out certain research question such as

1. How does our Hybrid feature selection strategy enhance the performance of SDP?
2. How local models enhance the performance over global models?

The remainder of the paper is coordinated as follows. In section 2, we examined the connected work about the utilization of GL models for defect prediction. Section 3 presents HFSS. Section 4 elaborates the experimental setup with algorithms to implement GL models with HFSS. Section 5 gives the exploratory outcomes and contrasts the outcomes with existing investigations. Then we present the lessons gained from analysis. Finally we conclude our findings in section 6.

## II. RELATED WORK

There has been enormous exploration in the space of SDP. In a study by J.Ostrand [6], they considered the heterogeneity of data to be the reason for conclusion instability in various studies. They observed that most studies don't have a conclusion about which method is best rather studies just classify the methods being best in different contexts. So they proposed the concept of global and local (GL) models for which they developed two algorithms namely "WHERE" to cluster the data and "WHICH" to learn local lessons from each cluster. Xu et.al. [8] finds the problem to keep data private while training models on private data distributed over several devices and issue of scalability in large models. So they proposed a federated learning approach which uses local representations on individual devices and global models across multiple devices.

Problem of heterogeneity of data has been a concern in multiple studies. To deal with this concern, S. Amasaki [9] used local model in a case study over social data resulting in no advantage in performance in comparison to global models. So proposed a hybrid approach using GL models i.e. Multivariate Adaptive Regression Splines (MARS). In other study Pan et.al. [10] applied metric selection to remove less informative metrics from the source project followed by metrics matching between source and target using methods of percentiles, Kolmogorov-Smirnov Test, Spearman's correlation.

However, the issues in defect prediction still exist. For instance, Menzies et al. discussed the issue of conclusion instability in software engineering research [19]. He states that studies do not provide a final conclusion regarding which technique or solution is best, and just classify the results based on the deployed techniques. They consider that conclusion instability is mainly caused by data heterogeneity. Their review uncovered that when similar strategies are applied globally to all information and when applied to intra-cluster information, and the outcomes are very unique. They found that local analysis has better results when the same methods are applied to GL data [19]. Feature selection, instance selection, extension of data collection and monte carlo simulation over space options have been effective in handling instability problems.

There is lack of studies which focus on local models for SDP. The local models were first introduced by Menzies et al. [19][18] for defect prediction and effort estimation, and later studied by Bettenburg et al.[20][21]. Both of them observe similar results for precision in defect prediction. In the third study, Scanniello[22] investigated the dependencies among the classes as criteria for clustering or grouping using borderFlow clustering algorithm rather than similar properties of classes.

Yang conducted an experiment to check validity of local models in JIT SDP context [11]. The experimental study was done on five open source software projects (OSSP) with three scenarios cross-validation, cross project validation, and time-wise cross-validation. They used k-medoids to split training data in several homogeneous regions for local models. Logistic Regression (LR) and effort aware linear regression (EALR) were used for classification models and effort aware predictor [11]. Their empirical results showed that local models performed worse than global models in classification. Another researcher [23]studied the impact of leading and trailing source lines of code changed in commit. They called it context metrics and carried out an empirical study on six open source projects for JIT defect prediction[23].Their context metrics outperformed the traditional code churn metrics with respect to the Area Under receiver operating characteristic Curve (AUC) and Metthews Correlation Coefficient. Zhang and colleagues proposed an effort aware Tri-training (EATT) semi-supervised method for JIT defect prediction[17]. Trautsch et al. proposed a cost saving model for JIT defect prediction by improving data collection, data labelling and feature selection. They considered the manually validated issues and improved Sliwerski-Zimmermann-Zeller (SZZ) algorithm for labelling the data and carried out a case study on thirty eight java projects[24].

Poncin developed a framework for analysing software repositories for combining different repositories and matching emails sent about a file, file modification and bug reports[25]. Thong [26] proposed a framework using deep learning to automatically extract features from code changes, commit messages to identify the defects and conducted experiments on QT and OPENSTACK. Aalok [27] studied the concept of sleeping defects. Sleeping defects are those which are not recognized in intermediate release but in later release. They analysed six open source projects from Github from the Apache ecosystem.

Finding a relation between commits and bugs has been a challenging task since the beginning of software defect prediction. SZZ is a heuristic based solution for finding defect inducing changes. Different researchers have fostered their own implementation of the SZZ algorithm. [28] presented an open implementation of SZZ for Github repositories. Fukushima [29] has proposed JIT models which provide eager feedback as compared to traditional models. They have used data from multiple software projects to produce training data and also integrate multiple models to create ensemble models. Jing et al. has provided a solution to class imbalance problems using Subclass Discriminant Analysis (SDA) [30]. SDA is a feature learning method, which divides a class into multiple subclasses and learns features from it using robust classification.

Borandag [7] proposed a feature selection algorithm using majority vote from multiple filters namely information gain, symmetrical uncertainty, Relief-F and correlation based filters. Shippey [2] utilized AST N-grams to identify the most encouraging features for software defect prediction. A cluster based approach FeSCH (Feature selection using Clusters of Hybrid data) [31] was proposed to extract the software features using the ranking strategy, feature selection ratio and classifiers for cross project defect prediction. Shivajikumar [32] empirically compared various attribute selection filter and wrapper method to obtain the feature set with optimal performance using SVM and Naive Bayes. They used scores from gain ratio attribute evaluation, chi-squared attribute evaluation, significance attribute evaluation, relief-F attribute selection and wrapper methods to select the features in an iterative manner until optimal performance is reached.

Motivation:

Menzies [19] found that, local models are superior to global models but Yang [11] contradicts their finding by presenting poor performance of local models in comparison to global models for classification. Pecorelli [33] presented no difference in performance of GL models. So it becomes significant to explore GL models in depth and present a tangible solution.

Data collected from repositories is not that clean to be directly consumable in predictive models. Supervised machine learning models suffers from various issue due to missing values, redundant data, irrelevant data, outliers, imbalanced and highly unrelated data. These pitfalls in data could be due to different scale of measurements for attributes, attribute types like Boolean or numeric, dependence among attributes like age is derived from date of birth, data collection issues. So attribute / feature selection play a crucial role in predictive performance of supervised machine learning algorithms as stated in [10] [31]. Jayanthi & Florence [4] used Principal Component Analysis (PCA) based feature reduction to improve accuracy of classification models. Borandag [7] proposed Majority vote based feature selection by first ranking the metrics on the basis of their importance and followed by voting scheme. Chao Ni [34] proposed two step feature selection method which includes, initial feature cluster selection based on density followed by ranking to choose appropriate cluster. Shivaji et. al. [32] utilizes various feature selection techniques such as Gain ratio, Chi-Squared, Relief-F, wrapper method using SVM and NB. So we proposed HFSS presented in next section.

Most studies consider the problem of heterogeneity, high variability in dataset and their solution include developing local models, multivariate adaptive regression splines and metrics selection to improve the performance of prediction models. In our study, we are using GL models in conjunction with HFSS for defect prediction. This study is first of its kind to the best of our knowledge.

## III. HYBRID FEATURE SELECTION STRATEGY (HFSS)

HFSS uses two methods of feature selection to prepare data for best predictions. First is univariate selection technique which uses statistical tests to find the features with best relationship with target class. In this technique, we use chi-squared statistical test-based technique to find relationship score of each attribute with the target attribute. Second is feature importance determination technique which uses extra tree classifier to give scores to each feature in dataset, higher the score more the relevance of feature for target attribute.

Chi-squared based selectKBest approach

We implemented our approach using the library functions from scikitlearn python library using selectKBest() function with chi-squared statistical method. It provides the score of relationship between features and the target class. Since chi-square is a univariate test so it does not consider the inter-relation between features. This is a kind of filter method which brings the top features with high relation with target feature. Feature score for all attributes of KC1 dataset are shown in Table 1.

Extra tree classifier (ETC)

Extra Tree classifier is ensemble learning based feature importance technique, which determine the importance of features by aggregating the results from multiple de-correlated decision trees. De-correlated decision tree uses random samples of n features from original dataset. Each decision tree gets the n number of features selected and information gain is calculated for all of the selected features. Information gain determines the reduction in entropy by transforming the dataset. Increase in information gain of a variable reduces the entropy and divide the data into multiple groups for effective classification.

In our case we are using a binary classification, so entropy can be calculated as

$$\text{Entropy} = -(p(0) * \log(P(0)) + p(1) * \log(P(1)))$$

Data with a 50/50 distribution of samples for the two groups would have maximum entropy, whereas an highly imbalanced dataset with a distribution of 20/80 would have smaller entropy for a randomly drawn sample from the data.

Information gain is used as the decision criteria to select the best features from dataset.

$IG(S, a) = H(S) – H(S \mid a)$

where $IG(S,a)$ is the information gain for dataset S for a random attribute a, $H(S)$ is the entropy of dataset before any transformation, $H(S|a)$ is the conditional entropy for given dataset with respect to attribute a, which can be calculated as

$H(S \mid a) = \text{sum } v \text{ in } a\ Sa(v)/S * H(Sa(v))$

where $Sa(v)/S$ is the proportion of the number of samples in the dataset with attribute a has the value v, and $H(Sa(v))$ is the entropy of each group of samples where attribute a has the value v.

Finally aggregating the information gain of each feature of dataset provides the ETC score. ETC score for KC1 dataset are depicted in Table 1. This can be accomplished utilizing the scikit-learn python libraries.

Table 1: Feature selection using SelectKBest and ETC of KC1

| Feature index | SelectKBest Score (A) | ETC Score (B) | Product (A*B) | Average of A & B |
|---|---|---|---|---|
| 0 | 16.515864 | 0.10062259 | 1.661869012 | 8.308243295 |
| 1 | 13.864425 | 0.03607222 | 0.500120589 | 6.95024861 |
| 2 | 4.850053 | 0.02296492 | 0.111381079 | 2.43650896 |
| 3 | 11.928183 | 0.03742913 | 0.446461512 | 5.982806065 |
| 4 | 13.951151 | 0.05930609 | 0.827388217 | 7.005228545 |
| 5 | 12.065444 | 0.06277575 | 0.757417296 | 6.064109875 |
| 6 | 11.653458 | 0.04660503 | 0.54310976 | 5.850031515 |
| 7 | 24.944794 | 0.05937611 | 1.481124832 | 12.50208506 |
| 8 | 15.896106 | 0.06095661 | 0.968972734 | 7.978531305 |
| 9 | 7.361797 | 0.05576291 | 0.410515224 | 3.708779955 |
| 10 | 12.364661 | 0.04277111 | 0.528850276 | 6.203716055 |
| 11 | 7.361727 | 0.05531765 | 0.407233438 | 3.708522325 |
| 12 | 16.276982 | 0.07621391 | 1.240532441 | 8.176597955 |
| 13 | 18.683501 | 0.05439372 | 1.016265122 | 9.36894736 |
| 14 | 19.307992 | 0.06350016 | 1.226060581 | 9.68574608 |
| 15 | 13.253962 | 0.05855625 | 0.776102312 | 6.656259125 |
| 16 | 15.128712 | 0.07149166 | 1.081576735 | 7.60010183 |
| 17 | 14.083211 | 0.03588416 | 0.505364197 | 7.05954758 |
|  |  | Mean | **0.805019** | **6.958112** |

Source: Authors, (2024).

In order to optimize the accuracy of our supervised machine learning models, we took two cases of feature selection depending upon the scores obtained using selectKBest and ETC. In first case we multiplied the score of selectKBest and ETC values while in second case we took the average of two scores. After calculating product and average of scores, it is critical to decide the number of attributes to be selected. For that we took mean of product and average scores and used it as minimum threshold value to determine the number of attributes to be chosen. So features selected as per threshold values shown in Table 1, for product metrics are 0, 7,12,14,16,13,8,4 and for average metrics are 7,14,13,0,12,8,16,17,4,1. So a total of eight features are selected using product metrics and ten features are selected using average metrics.

## IV. EXPERIMENTAL SETUP

In this section, we first describe the datasets used in our experiment. Then we introduce the models selected to carry out defect prediction. We provide the details of our empirical setup, which enhances the software defect prediction as compared to traditional techniques.

Dataset: We applied our proposed approach to existing benchmark dataset from PROMISE repository. PROMISE dataset was developed by Jureczko et. al. [35] containing defect data from java projects. PROMISE repository datasets are downloaded from the internet in .csv format. Selected projects from PROMISE are JM1, PC3 and KC1. Since all these software projects dataset contain some extra attributes in comparison to software metrics used by our framework. So removing the extra features make this data directly usable for our framework. JM1, PC3 and KC1 contains 10885, 1563, 2109 rows respectively. Detailed statistics of dataset are shown in Table 2.

Table 2: Statistics of dataset

| Project | Number.of rows | Numbers of attributes | % buggy | Duplicates entries |
|---|---|---|---|---|
| KC1 | 2109 | 22 | 15.45 | 900 |
| JM1 | 10885 | 22 | 19.35 | 0 |
| PC3 | 1563 | 38 | 10.23 | 157 |

Source: Authors, (2024).

**Data Pre-processing:** Data pre-processing plays a critical role in order to build high performing prediction models. Our system deploys five steps of data pre-processing.

1. **Duplicate Removal**: In this step all the duplicate entries are removed while keeping only the first entry.
2. **Handling missing values**: The data contained some blank or empty fields. Their values might be either optional or left out due to developer's negligence. The mean of the attribute method was used to fill those gaps.
3. **Scaling**: Scaling of data becomes important when distance based modelling is used rather than tree based. So normalization of data is a required for SVM, KNN and Naive Bayes while not for Random Forest.
4. **Managing Outliers:** The outliers in the data have a tremendous impact on the predictive power. There are a number of outliers handling techniques like z-score, IQR, removal of outliers etc. This study used the z-score >3 as a threshold to mitigate outliers in our dataset.
5. **Feature selection**: When attributes of a dataset are tightly coupled, leading to overfitting and underfitting problems. Then it becomes obvious to reduce dimensions by selection of relevant features only. Our hybrid feature selection technique discussed in section 3 is used for feature selection. Features selected for each data will be different since the feature scores obtained using our approach are calculated dynamically.

Model selection: In our experiment, we applied four machine learning models to measure the efficiency of our proposed approach. Most machine learning algorithm suffers from the problem of either overfitting or underfitting. Overfitting refers to a model that models the training data excessively well while underfitting refers to a data model that cannot train a model properly and cannot generalize to new data. Both of these lead to poor performance of models. Overfitting is more probable with non-parametric and non-linear models that have greater adaptability when learning a target function. We applied three non-parametric models KNN, Random Forest (RF), SVM and one parametric model Naive Bayes (NB). There are two reasons for choosing these models for defect prediction. First these models are widely used in research community and have proven their ability. Secondly there exist some underlying differences in their implementation while performance is quite comparable with other baselines studies. All these models have been utilized with their default parameter setting using scikit learn python libraries, which make our experiment effectively reproducible.

**Model building:** Our approach deploys two types of models, namely global models and local models. Global models use complete available data from a given dataset to build prediction models, while internally partitioning it into training and test sets.

**Algorithm for Global Models:**

```
1. IMPORT required libraries including Scikit learn
2. IMPORT the dataset in .csv format
3. PRE-PROCESSING
     a. Remove duplicates from data while keeping the first row.
     b. Replace missing values with mean of attribute values.
     c. REMOVE Outliers using z-score values < 3
     d. FEATURE SELECTION using Hybrid Feature selection Strategy [section 3]
4. SCALE data using MinMaxScaler
5. Split data into training and test set in 70:30 ratio.
6. FOREACH model RF, SVM, KNN and NB
     a. Train classifier model using training set.
     b. Calculate performance metrics (accuracy, precision, recall, f-measure) of model using available test set.
   END FOREACH
```

In Local models, whole of the data is first splitted into homogeneous regions and then individual classification algorithms are applied to individual regions. These models help in handling problems of heterogeneity in data [14]. Herbold et.al. [14] evaluated the local models and investigated their advantages and drawbacks in cross project scenarios. They also compared the local models with global models and transfer learning models in cross project. They found negative improvement using the local model in comparison to global models.

In our system, we initially partitioned data into different regions in local models, but in later phase local models are similar to global models. Hence deploy the machine learning models to create training and testing sets before finally building the model. Our algorithm uses k-means to create clusters. Since dataset size of our most projects is in thousands only, so we selected k=2, so that each of the cluster contains some data values.

**Algorithm for Local Models:**

```
1. IMPORT required libraries including Scikit learn
2. IMPORT the dataset in .csv format
3. PRE-PROCESSING
     a. Remove duplicates from data while keeping the first row.
     b. Replace missing values with mean of attribute values.
     c. REMOVE Outliers using z-score values < 3
     d. FEATURE SELECTION using Hybrid Feature selection Strategy [section 3]
4. SCALE data using MinMaxScaler
5. CREATE CLUSTERS(C1 and C2) using k-means clustering with k=2
6. FOREACH CLUSTER C1 and C2
     a. Drop target column from dataset.
     b. Split data into training and test set in 70:30 ratio.
     c. FOREACH model RF, SVM, KNN and NB
          i. Train classifier model using training set.
          ii. Calculate performance metrics (accuracy, precision, recall, f-measure) of model using available test set.
          iii. END FOREACH
END FOREACH
```

Table 3: Data dimensions of the experiment.

| Project | Model type | Cluster | Dataset Size | After Duplicate Removal | Training Size | Test Size |
|---|---|---|---|---|---|---|
| JM1 | Local | C1 | (10885, 19) | (8883, 19) | (1192, 10) | (511, 10) |
| JM1 | Local | C2 | (10885, 19) | (8883, 19) | (4601, 10) | (1973, 10) |
| JM1 | Global | - | (10885, 19) | (8883, 19) | (5793, 18) | (2484, 18) |
| KC1 | Global | - | (2109, 19) | (1209, 19) | (775, 18) | (333, 18) |
| KC1 | Local | C1 | (2109, 19) | (1209, 19) | (531, 10) | (228, 10) |
| KC1 | Local | C2 | (2109, 19) | (1209, 19) | (244, 10) | (105, 10) |
| PC3 | Local | C1 | (1563, 19) | (1406, 19) | (574, 10) | (246, 10) |
| PC3 | Local | C2 | (1563, 19) | (1406, 19) | (347, 10) | (150, 10) |
| PC3 | Global | - | (1563, 19) | (1406, 19) | (921, 18) | (396, 18) |

Source: Authors, (2024).

Table 3 showcase the dataset dimension of all three projects after cleaning of dataset to remove duplicates and division into training and testing set for machine learning models.

**Evaluation measure:** A number of performance evaluation methods are available for machine learning classifiers. We use four performance indicators i.e. Accuracy, Precision, Recall and f-measure. These are most widely used in research community which make comparison of results easy with other baseline studies. Accuracy is an indicator of how many classifications are correct. Precision and recall contradict each other, so keeping balance of both is required. Higher the recall will lead to low precision. While F-measure is a harmonic mean of recall and precision, which give the most accurate measure of performance of predictive model.

**Baseline methods:** To prove the effectiveness of our approach we compared our HFSS with two baseline studies for defect prediction techniques i.e. DPDF [3], Neural Network classifier based approach [4]. In their study Zhou et. al. proved that their deep forest based model for software defect prediction using cascade multilayer ensemble of random forest has better performance than gcForest. They compared DPDF using 25 projects from NASA, PROMISE, AEEEM and Relink datasets. They also used z-score to process the original data. Jayanthi et. al. [4] applied improved version of PCA using artificial neural network to establish the performance of their models on 4 projects from PROMISE dataset. Since their study used artificial neural network (ANN) and they have provided no nomenclature to their technique, so we will denote their technique as “ANN” in our comparison. Both of these studies use accuracy, precision, recall and f-measure as common measure of performance.

In RQ 1, we evaluate HFSS impact on performance of our machine learning models using RF, SVM, KNN and NB. HFSS deploy two methods to generate importance score for each attribute from dataset, so feature selection process is totally dynamic and features selected for each dataset can vary since only highly scoring top ten features are selected. More the target feature relies on any attribute better will be prediction. We applied HFSS on GL models using RF, SVM, KNN and NB.

In RQ 2, we evaluate the effect of heterogeneity of data on the performance of machine learning models using RF, SVM, KNN and NB. A local model uses internal clustering to find the non-heterogeneous region of data that exist within dataset. Global models uses complete data as single unit which is bifurcated into training set and testing set while in local models each cluster is further bifurcated into training and testing set so resulting in quite smaller dataset. Both GL models also use our hybrid feature selection strategy to further upscale the prediction performance of software defect prediction.

We performed our experiment using Intel (R) Core(TM) i5 -7200U CPU @ 2.50GHz, 8GB RAM. The programming language for scripting is Python 3.7 installed in Windows 10 operating system.

## V. RESULTS AND DISCUSSIONS

In this section, we provide the empirical results from our study to answer the research questions.

RQ 1: How does our Hybrid feature selection strategy enhance the performance of SDP?

To the best of our knowledge, we are the first one to propose a hybrid feature selection strategy using a univariate selection technique using chi-square statistical test and ensemble learning based feature importance technique to find the best possible features to give highly performing software defect prediction results. We provide the empirical results for baseline datasets using HFSS. In this research question we are concerned with only hybrid feature selection approach, so we will be comparing our global models using hybrid approach with baselines results. In order to make the understanding easier, we have highlighted the best results in tabular data by making them bold.

Table 4: Experimental results for project KC1 using average of features score

| **Project** | **Model** | **Accuracy** | | | **Recall** | | | **Precision** | | | **f-measure** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Global | Local | | Global | Local | | Global | Local | | Global | Local | |
| | | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 |
| **KC1** | RF | 0.89 | 0.93 | 0.70 | 0.98 | 0.99 | 0.86 | 0.90 | 0.93 | 0.74 | 0.94 | 0.96 | 0.79 |
| | SVM | **0.89** | 0.92 | 0.69 | **0.98** | 1.00 | 0.97 | **0.90** | 0.92 | 0.69 | **0.94** | 0.96 | 0.81 |
| | NB | 0.89 | 0.80 | 0.59 | 0.98 | 0.82 | 0.73 | 0.90 | 0.96 | 0.69 | 0.94 | 0.88 | 0.71 |
| | KNN | 0.89 | 0.92 | 0.68 | 0.98 | 1.00 | 0.98 | 0.90 | 0.92 | 0.68 | 0.94 | 0.96 | 0.81 |
| | ANN | 0.87 | | | **0.98** | | | 0.87 | | | 0.92 | | |

Source: Authors, (2024).

Table 5: Experimental results for project KC1 using product of features score

| **Project** | **Model** | **Accuracy** | | | **Recall** | | | **Precision** | | | **f-measure** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Global | Local | | Global | Local | | Global | Local | | Global | Local | |
| | | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 |
| **KC1** | RF | 0.89 | 0.93 | 0.68 | 0.98 | 0.99 | 0.88 | 0.91 | 0.94 | 0.71 | 0.94 | 0.96 | 0.78 |
| | SVM | **0.89** | 0.93 | 0.67 | **0.98** | 1.00 | 0.98 | **0.91** | 0.93 | 0.67 | **0.94** | 0.96 | 0.80 |
| | NB | 0.89 | 0.82 | 0.56 | 0.98 | 0.85 | 0.72 | 0.91 | 0.95 | 0.65 | 0.94 | 0.90 | 0.68 |
| | KNN | 0.89 | 0.93 | 0.66 | 0.98 | 1.00 | 0.99 | 0.91 | 0.93 | 0.66 | 0.94 | 0.96 | 0.79 |
| | ANN | 0.87 | | | **0.98** | | | 0.87 | | | 0.92 | | |

Source: Authors, (2024).

In Table 4, we have results of our model using features selected by taking average of feature score. It can be seen that we have only one baseline study named ANN, accuracy of ANN is lower than accuracy of our models using RF, SVM, NB and KNN. In case of recall, our global models have equal results than that of ANN. But in case of precision and f-measure our models lead with 3% and 2% respectively in comparison to ANN. In Table 5, we have results of our models using features selected by taking product of feature scores. In this case results obtained by our models are higher than baseline study. But the results from this experiment are even higher than the results from earlier experiment. Table 4 and Table 5 results for our global models are same in case of accuracy, recall and f-measure but 1% higher precision when product of features score is used as selection critieria.

Table 6 Experimental results for project JM1 using average of features score

| **Project** | **Model** | **Accuracy** | | | **Recall** | | | **Precision** | | | **f-measure** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Global | Local | | Global | Local | | Global | Local | | Global | Local | |
| | | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 |
| **JM1** | RF | 0.81 | 0.78 | 0.91 | 0.96 | 0.96 | 0.97 | 0.83 | 0.80 | 0.94 | 0.89 | 0.87 | 0.95 |
| | SVM | 0.81 | 0.78 | 0.92 | 0.96 | 0.99 | 1.00 | 0.83 | 0.78 | 0.92 | 0.89 | 0.88 | 0.96 |
| | NB | 0.81 | 0.75 | 0.87 | 0.96 | 0.88 | 0.95 | 0.83 | 0.81 | 0.92 | 0.89 | 0.85 | 0.93 |
| | KNN | 0.81 | 0.78 | 0.92 | 0.96 | 0.99 | 1.00 | 0.83 | 0.79 | 0.92 | 0.89 | 0.88 | 0.96 |
| | DPDF | 0.79 | | | 0.15 | | | 0.49 | | | 0.23 | | |
| | ANN | **0.83** | | | **0.98** | | | **0.83** | | | **0.90** | | |

Source: Authors, (2024).

Table 7 Experimental results for project JM1 using product of features score

| **Project** | **Model** | **Accuracy** | | | **Recall** | | | **Precision** | | | **f-measure** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Global | Local | | Global | Local | | Global | Local | | Global | Local | |
| | | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 |
| **JM1** | RF | 0.81 | 0.79 | 0.90 | 0.96 | 0.95 | 0.97 | 0.83 | 0.82 | 0.92 | 0.89 | 0.88 | 0.95 |
| | SVM | 0.81 | 0.80 | 0.90 | 0.96 | 0.99 | 1.00 | 0.83 | 0.80 | 0.90 | 0.89 | 0.89 | 0.95 |
| | NB | 0.81 | 0.76 | 0.13 | 0.96 | 0.87 | 0.03 | 0.83 | 0.83 | 0.94 | 0.89 | 0.85 | 0.07 |
| | KNN | 0.81 | 0.79 | 0.90 | 0.96 | 0.98 | 1.00 | 0.83 | 0.80 | 0.90 | 0.89 | 0.88 | 0.95 |
| | DPDF | 0.79 | | | 0.15 | | | 0.49 | | | 0.23 | | |
| | ANN | **0.83** | | | **0.98** | | | **0.83** | | | **0.90** | | |

Source: Authors, (2024).

In Table 6 and Table 7, we have our results of using machine learning models on JM1 dataset, for this dataset we have one more baseline study DPDF [3]. Our global models for JM1 fail to surpass the performance of ANN in terms of accuracy, precision, recall and f-measure but our models performed better than the DPDF, which have 2% lower accuracy, while precision, recall and f-measure of DPDF are very low at 15%, 49% and 23% respectively. For JM1, both of our feature selection criteria have produced same results.

Table 8 Experimental results for project PC3 using average of features score

| **Project** | **Model** | **Accuracy** | | | **Recall** | | | **Precision** | | | **f-measure** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Global | Local | | Global | Local | | Global | Local | | Global | Local | |
| | | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 |
| **PC3** | RF | 0.91 | 0.93 | 0.88 | 0.98 | 0.99 | 0.97 | 0.93 | 0.93 | 0.90 | 0.95 | 0.96 | 0.93 |
| | SVM | 0.91 | 0.93 | 0.88 | 0.98 | 1.00 | 1.00 | **0.93** | 0.93 | 0.88 | 0.95 | 0.97 | 0.94 |
| | NB | **0.91** | 0.89 | 0.82 | 0.98 | 0.93 | 0.88 | 0.93 | 0.95 | 0.91 | 0.95 | 0.94 | 0.90 |
| | KNN | 0.91 | 0.93 | 0.88 | 0.98 | 1.00 | 1.00 | 0.93 | 0.93 | 0.88 | 0.95 | 0.97 | 0.94 |
| | DPDF | 0.90 | | | 0.07 | | | 0.26 | | | 0.11 | | |
| | ANN | 0.90 | | | **0.99** | | | 0.90 | | | **0.95** | | |

Source: Authors, (2024).

Table 9 Experimental results for project PC3 using product of features score

| **Project** | **Model** | **Accuracy** | | | **Recall** | | | **Precision** | | | **f-measure** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Global | Local | | Global | Local | | Global | Local | | Global | Local | |
| | | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 | G | C1 | C2 |
| **PC3** | RF | 0.91 | 0.96 | 0.85 | 0.98 | 1.00 | 0.96 | 0.93 | 0.96 | 0.88 | 0.95 | 0.98 | 0.92 |
| | SVM | 0.91 | 0.96 | 0.86 | 0.98 | 1.00 | 1.00 | 0.93 | 0.96 | 0.86 | 0.95 | 0.98 | 0.92 |
| | NB | **0.91** | 0.93 | 0.79 | 0.98 | 0.97 | 0.85 | **0.93** | 0.97 | 0.91 | 0.95 | 0.97 | 0.88 |
| | KNN | 0.91 | 0.96 | 0.86 | 0.98 | 1.00 | 1.00 | 0.93 | 0.96 | 0.86 | 0.95 | 0.98 | 0.92 |
| | DPDF | 0.90 | | | 0.07 | | | 0.26 | | | 0.11 | | |
| | ANN | 0.90 | | | **0.99** | | | 0.90 | | | **0.95** | | |

Source: Authors, (2024).

In Table 8 and Table 9, we presented our results of experiment based on the features selected using average and product of feature score obtained by hybrid feature selection strategy for PC3 dataset. It can be observed in the tables that, our models outperformed in terms of accuracy and precision when compared to ANN and DPDF while in case of recall and f-measure ANN has better performance. Performance of models is same for both types of feature selection criteria which lead us to experiment more with the criteria of setting the final score of feature selection.

Since all three dataset has different metrics set, we applied our hybrid approach individually to grab the most important features out of the total metrics set. In comparison to baselines studies DPDF [3] and ANN [4], our prediction results in terms of accuracy, precision, recall and f-measure are highly significant. Since our dataset share only two common dataset with DPDF and three common dataset between ANN. Our results when compared with DPDF are 10% higher for JM1, 6% more for PC3 dataset in terms of accuracy. While results for precision, recall and f-measure using our technique are above 90% for each dataset, while DPDF results are below 50%. So there is a huge improvement in performance using hybrid feature selection strategy. When compared with ANN, for KC1 dataset our accuracy, precision, recall and f-measure are 6%, 2%, 8% and 4% higher respectively. For JM1 dataset, our accuracy, precision, recall and f-measure are 7%, 2%, 7% and 5% higher respectively. For PC3 dataset, our accuracy, precision, recall and f-measure are 6%, 1%, 7% and 3% higher respectively.

Reasons for our approach to achieve better performance are as follows:1.

Compared with traditional machine learning approach and approach considered as baseline, our approach uses a more robust hybrid feature selection strategy which makes our models perform better by utilizing the best set of features and dynamic selection of features by calculating the importance score of each attribute on the fly:2.

Initially we adopted the process to find the most importance by taking mean of the scores obtained using product and average. But using mean of scores as threshold resulted in quite less number of features to be selected for model building and thereby resulting in poor performance. So we analysed the results from multiple iteration of experiment that the performance of all those models was satisfactory where number of features selected are more than 8 and less than 12. Finally it was observed that the models with 10 features selection have the majority and best performance. So we set the number of features to be selected be top 10 on the basis of scores obtained using product and average.

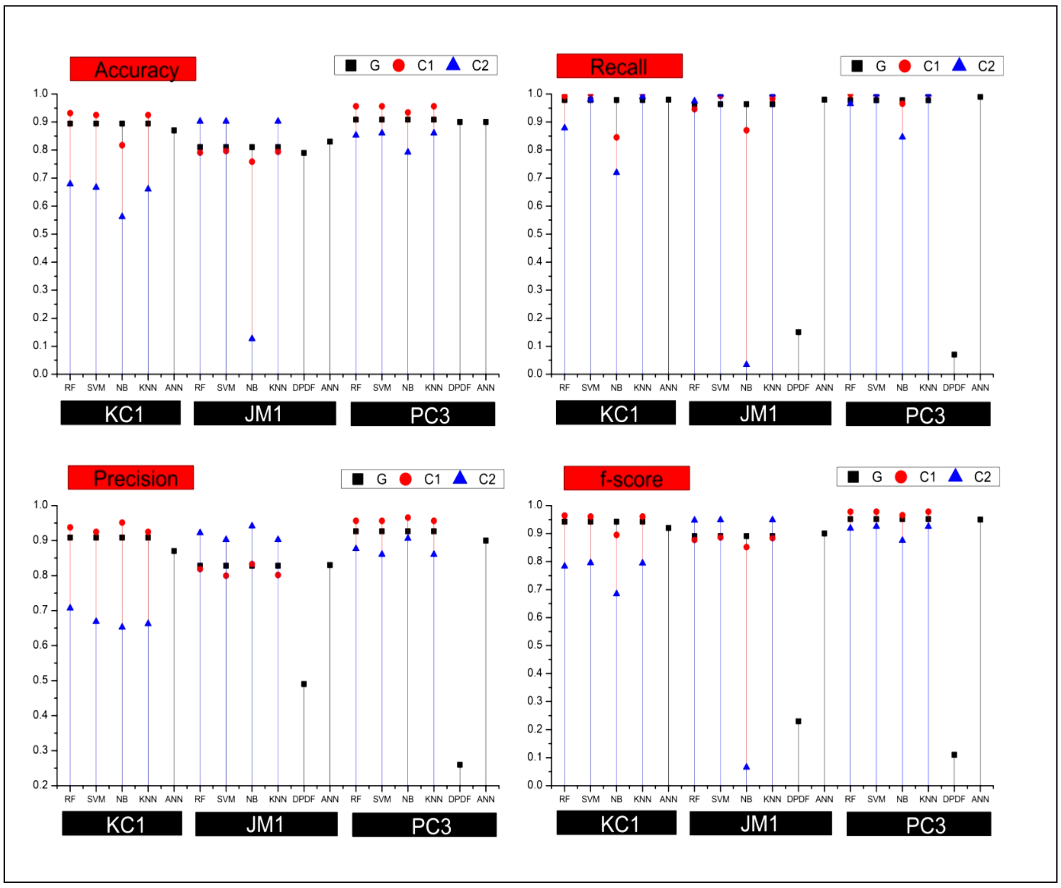


Figure:2 Dropline representation of empirical results using GL model with hybrid feature selection approach based on product.
Source: Authors, (2024).

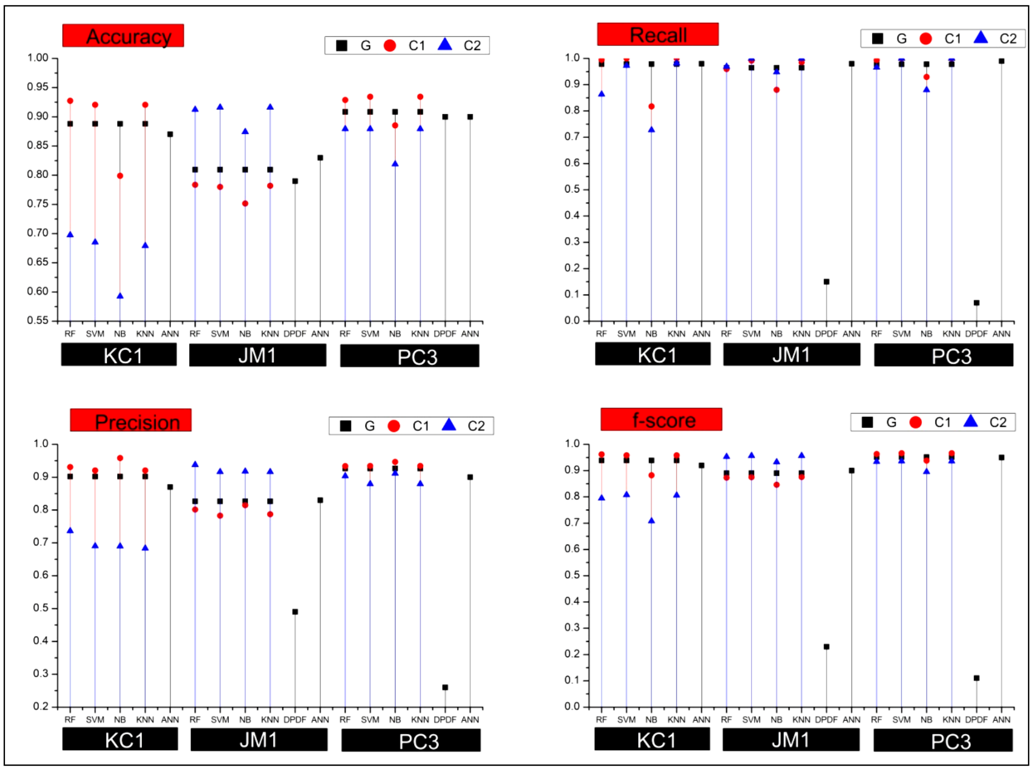


Figure:3 Dropline representation of empirical results using GL model with hybrid feature selection approach based on average.
Source: Authors, (2024).

Additional inferences can be deduced by comparing the dropline graphs shown in Fig. 2 and Fig. 3, which showcase the performance of our approach with different criteria of feature selection using hybrid feature selection approach. Product criteria of feature selection have attained better performance than average criteria for both GL models in majority of projects as well as performance metrics such as accuracy, precision, recall and f-measure.

RQ 2: How local models enhance the performance over global models?

All traditional models as well as the baseline models uses complete dataset to generate training and testing set for their deep forest and neural network, which suffers from the problem of heterogeneity, our local models handle this problem by creating homogeneous clusters of data internally. So each local models creates two clusters C1 and C2, then further process to build models is similar to global models.

Results of KC1 shown in **Erro! Argumento de opção desconhecido.** depicts that accuracy of our global models is higher than the baseline study ANN because of hybrid feature selection strategy using average of features scores. This performance is further enhanced by using local models and it is evident from the results that local models score higher accuracy than that of global model, but it is true for only C1 cluster. Cluster C2 has poor performance as compared with its peer cluster C1, global and baseline ANN. These trends are repeated for precision, recall and f-measure too. In Table 5, we have prediction performance of our local models using hybrid feature selection strategy using product of feature scores. The results are better when compared to criteria of average feature score.

**Erro! Argumento de opção desconhecido.** depicts the result of dataset JM1, global models has poor accuracy as compared to ANN, while our local models have better accuracy than ANN. But only one cluster C2 has obtained better accuracy, C1 has accuracy lower than global models too. In case of precision, recall and f-measure global models have poor performance as compared to ANN but much higher than DPDF. Similar to accuracy, all of the local models of cluster C2 have best performance of all but C1 being least performing of all. In Table 7, we have results from criteria of selection as product of feature scores. Similar pattern of results can be observed when compared to criteria of average of features scores. When compared side by side, cluster C1 and C2 in Table 6 and Table 7, results in later case are comparatively better.

Table 8 and Table 9 shows the empirical results of dataset PC3, our global models have outperformed in terms of accuracy and precision when compared with baseline studies ANN and DPDF. But they fail to outperform the baseline ANN in terms of recall while resulting in equal performance in terms of f-measure. Till now for project KC1 and JM1 we have observed that local models have performed better than our global models. Same is the case with this project; local models have better results in C1 and poor results in cluster C2. Following the patterns observed in project KC1 and JM1, our feature selection criteria of product of feature scores have always performed better the criteria of average of features score.

For each project, we get one observation of accuracy, precision, recall and f-measure for global models and 2 observations for cluster C1 and C2 of local models. A total of 12 observations are obtained for each machine learning model corresponding to each project. So we have a total of 144 observations with 48 observations per projects. For global models all of the machine learning models (SVM, NB, RF and KNN) have equal performance in terms of accuracy, precision, recall and f-measure. SVM and NB have dominance in at least one of performance metrics results of cluster C1 and C2 of local models. SVM has highest accuracy in all three projects in cluster C1 defeating the baseline results. SVM also has highest recall and f-measure for all three projects with two highest in cluster C1 and one highest in C2. NB has highest precision for all three projects with leading twice in cluster C1 and once in C2. From the above facts obtained from empirical study, we can deduce that SVM and NB based machine learning models are sufficient to build efficient defect prediction system. SVM is non-parametric model and NB is parametric model, so they contradict each other and can be effectively used to balance the performance.

Overall by observing the results for three projects KC1, JM1 and PC3 as represented in Fig. 2, we can conclude that local models in conjunction with our hybrid feature selection strategy have proven to be most effective in terms of accuracy, precision, recall and f-measure than global models as well as the two baseline studies. We can analyse that results are quite different for cluster C1 and C2. This is because most of our dataset have imbalance data which results in one of the cluster being highly homogeneous and other cluster being left with the outliers and heterogeneous dataset and reduced dataset size. We call this as one-cluster problem where one cluster fails to obtain sufficient data due to grouping of majority in other cluster. Reduction is size has negative impact on performance when combined with heterogeneous data.

## VI. CONCLUSION

This study aims to boost the performance of SDP techniques by investigating the problem of heterogeneity within dataset using GL models. Local models provide the basis for mitigation of problem of heterogeneity by identifying internal regions of homogeneous data. We used four machine learning algorithms namely RF, SVM, KNN and NB for implementing our proposed approach and tested the validity of Hybrid Feature Selection Strategy (HFSS). We computed the performance of our approach in terms of accuracy, precision, recall and f-measure.

According to empirical study conducted on three projects of PROMISE repository, results indicate that our proposed approach using local models in conjunction with HFSS has achieved better results than baseline studies. Local models have proven to be convincing in obtaining homogeneous clusters of data. Certain local models suffered due to minimum data requirements when majority of data is group in one cluster while other cluster containing only outliers leading to one-cluster problem. In summary, our results show that our approach is practically feasible and can be applied on any datasets. In future, we will apply our approach to more datasets and try to provide solution to one-cluster problem.

## VI. AUTHOR'S CONTRIBUTION

**Conceptualization:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Methodology:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Investigation:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Discussion of results:** Pradip Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Writing – Original Draft:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Visualization, Writing, Editing:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Resources:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Supervision:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.
**Approval of the final text:** Vikas Suhag, Sanjay Kumar Dubey and Bhupendra Kumar Sharma.

## VII. REFERENCES

[1] D. Paterson, J. Campos, R. Abreu, G. M. Kapfhammer, G. Fraser, and P. McMinn, "An Empirical Study on the Use of Defect Prediction for Test Case Prioritization," in 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), Apr. 2019, pp. 346–357, doi: 10.1109/ICST.2019.00041.

[2] T. Shippey, D. Bowes, and T. Hall, "Automatically identifying code features for software defect prediction: Using AST N-grams," Inf. Softw. Technol., vol. 106, pp. 142–160, Feb. 2019, doi: 10.1016/j.infsof.2018.10.001.

[3] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest," Inf. Softw. Technol., vol. 114, no. July, pp. 204–216, 2019, doi: 10.1016/j.infsof.2019.07.003.

[4] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," Cluster Comput., vol. 22, no. S1, pp. 77–88, Jan. 2019, doi: 10.1007/s10586-018-1730-1.

[5] D. Rodriguez, J. Dolado, J. Tuya, and D. Pfahl, "Software defect prediction with zero-inflated Poisson models," pp. 2–5, Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.13717.

[6] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Predicting the location and number of faults in large software systems IEEE Transactions on Software Engineering, 31 (2005)," Pp., vol. 340, no. 4, pp. 340–355, 2005.

[7] E. Borandag, A. Ozcift, D. Kilinc, and F. Yucalar, "Majority vote feature selection algorithm in software fault prediction," Comput. Sci. Inf. Syst., vol. 16, no. 2, pp. 515–539, 2019, doi: 10.2298/CSIS180312039B.

[8] X. Yu, M. Wu, Y. Jian, K. E. Bennin, M. Fu, and C. Ma, "Cross-company defect prediction via semi-supervised clustering-based data filtering and MSTrA-based transfer learning," Soft Comput., vol. 22, no. 10, pp. 3461–3472, 2018, doi: 10.1007/s00500-018-3093-1.

[9] S. Amasaki, "On Applicability of Cross-project Defect Prediction Method for Multi-Versions Projects," in Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE, 2017, pp. 93–96, doi: 10.1145/3127005.3127015.

[10] C. Pan, M. Lu, B. Xu, and H. Gao, "An improved CNN model for within-project software defect prediction," Appl. Sci., vol. 9, no. 10, pp. 1–28, 2019, doi: 10.3390/app9102138.

[11] X. Yang, H. Yu, G. Fan, K. Shi, and L. Chen, "Local versus Global Models for Just-In-Time Software Defect Prediction," Sci. Program., vol. 2019, pp. 1–13, Jun. 2019, doi: 10.1155/2019/2384706.

[12] W. Y. Chen, Y. C. F. Wang, Y. C. Liu, Z. Kira, and J. Bin Huang, "A closer look at few-shot classification," 7th Int. Conf. Learn. Represent. ICLR 2019, no. 2018, pp. 1–17, 2019.

[13] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," Inf. Softw. Technol., vol. 54, no. 3, pp. 248–256, 2012, doi: 10.1016/j.infsof.2011.09.007.

[14] S. Herbold, A. Trautsch, and J. Grabowski, "Global vs. local models for cross-project defect prediction," Empir. Softw. Eng., vol. 22, no. 4, pp. 1866–1902, Aug. 2017, doi: 10.1007/s10664-016-9468-y.

[15] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous Defect Prediction," IEEE Trans. Softw. Eng., vol. 44, no. 9, pp. 874–896, Sep. 2018, doi: 10.1109/TSE.2017.2720603.

[16] Y. Li, Z. Huang, Y. Wang, and B. Fang, "Evaluating data filter on cross-project defect prediction: Comparison and improvements," IEEE Access, vol. 5, pp. 25646–25656, 2017, doi: 10.1109/ACCESS.2017.2771460.

[17] W. Li, W. Zhang, X. Jia, and Z. Huang, "Effort-Aware semi-Supervised just-in-Time defect prediction," Inf. Softw. Technol., vol. 126, no. June, p. 106364, 2020, doi: 10.1016/j.infsof.2020.106364.

[18] T. Menzies et al., "Local versus global lessons for defect prediction and effort estimation," IEEE Trans. Softw. Eng., vol. 39, no. 6, pp. 822–834, 2013, doi: 10.1109/TSE.2012.83.

[19] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. Cok, "Local vs. global models for effort estimation and defect prediction," 2011 26th IEEE/ACM

Int. Conf. Autom. Softw. Eng. ASE 2011, Proc., pp. 343–351, 2011, doi: 10.1109/ASE.2011.6100072.

[20] N. Bettenburg, M. Nagappan, and A. E. Hassan, “Think locally, act globally: Improving defect and effort prediction models,” IEEE Int. Work. Conf. Min. Softw. Repos., pp. 60–69, 2012, doi: 10.1109/MSR.2012.6224300.

[21] N. Bettenburg, M. Nagappan, and A. E. Hassan, “Towards improving statistical modeling of software engineering data: think locally, act globally!,” Empir. Softw. Eng., vol. 20, no. 2, pp. 294–335, 2015, doi: 10.1007/s10664-013-9292-6.

[22] G. Scanniello, C. Gravino, A. Marcus, and T. Menzies, “Class level fault prediction using software clustering,” 2013 28th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2013 - Proc., pp. 640–645, 2013, doi: 10.1109/ASE.2013.6693126.

[23] M. Kondo, D. M. German, O. Mizuno, and E. H. Choi, “The impact of context metrics on just-in-time defect prediction,” Empir. Softw. Eng., vol. 25, no. 1, pp. 890–939, 2020, doi: 10.1007/s10664-019-09736-3.

[24] A. Trautsch, S. Herbold, and J. Grabowski, “Static source code metrics and static analysis warnings for fine-grained just-in-time defect prediction,” Proc. - 2020 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2020, pp. 127–138, 2020, doi: 10.1109/ICSME46990.2020.00022.

[25] W. Poncin, A. Serebrenik, and M. Van Den Brand, “Process mining software repositories,” Proc. Eur. Conf. Softw. Maint. Reengineering, CSMR, pp. 5–13, 2011, doi: 10.1109/CSMR.2011.5.

[26] T. Hoang, H. Khanh Dam, Y. Kamei, D. Lo, and N. Ubayashi, “DeepJIT: An End-to-End Deep Learning Framework for Just-in-Time Defect Prediction,” in 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), May 2019, vol. 2019-May, pp. 34–45, doi: 10.1109/MSR.2019.00016.

[27] A. Ahluwalia, D. Falessi, and M. Di Penta, “Snoring: A noise in defect prediction datasets,” IEEE Int. Work. Conf. Min. Softw. Repos., vol. 2019-May, pp. 63–67, 2019, doi: 10.1109/MSR.2019.00019.

[28] M. Borg, O. Svensson, K. Berg, and D. Hansson, “SZZ unleashed: an open implementation of the SZZ algorithm - featuring example usage in a study of just-in-time bug prediction for the Jenkins project,” in Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation - MaLTeSQuE 2019, 2019, pp. 7–12, doi: 10.1145/3340482.3342742.

[29] T. Fukushima et al., An empirical study of just-in-time defect prediction using cross-project models, vol. 21, no. 5. Empirical Software Engineering, 2014.

[30] 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 Trans. Softw. Eng., vol. 43, no. 4, pp. 321–339, 2017, doi: 10.1109/TSE.2016.2597849.

[31] 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,” J. Comput. Sci. Technol., vol. 32, no. 6, pp. 1090–1107, 2017, doi: 10.1007/s11390-017-1785-0.

[32] S. Shivaji, E. James Whitehead, R. Akella, and S. Kim, “Reducing features to improve code change-based bug prediction,” IEEE Trans. Softw. Eng., vol. 39, no. 4, pp. 552–569, 2013, doi: 10.1109/TSE.2012.43.

[33] F. Pecorelli and D. Di Nucci, “Adaptive selection of classifiers for bug prediction: A large-scale empirical analysis of its performances and a benchmark study,” Sci. Comput. Program., vol. 205, p. 102611, May 2021, doi: 10.1016/j.scico.2021.102611.

[34] 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,” J. Comput. Sci. Technol., vol. 32, no. 6, pp. 1090–1107, 2017, doi: 10.1007/s11390-017-1785-0.

[35] M. Jureczko and L. Madeyski, “Towards identifying software project clusters with regard to defect prediction,” ACM Int. Conf. Proceeding Ser., 2010, doi: 10.1145/1868328.1868342.