electronics

Article

Optimal Feature Selection through Search-Based Optimizer in Cross Project

Rizwan bin Faiz [1], Saman Shaheen [1], Mohamed Sharaf [2] and Hafiz Tayyab Rauf [3,*]

[1] Faculty of Computing, Riphah International University, I-14 Campus Islamabad, Islamabad 46000, Pakistan
[2] Industrial Engineering Department, College of Engineering, King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia
[3] Centre for Smart Systems, AI and Cybersecurity, Staffordshire University, Stoke-on-Trent ST4 2DE, UK
* Correspondence: hafiztayyabrauf093@gmail.com

Abstract: Cross project defect prediction (CPDP) is a key method for estimating defect-prone modules of software products. CPDP is a tempting approach since it provides information about predicted defects for those projects in which data are insufficient. Recent studies specifically include instructions on how to pick training data from large datasets using feature selection (FS) process which contributes the most in the end results. The classifier helps classify the picked-up dataset in specified classes in order to predict the defective and non-defective classes. The aim of our research is to select the optimal set of features from multi-class data through a search-based optimizer for CPDP. We used the explanatory research type and quantitative approach for our experimentation. We have F1 measure as our dependent variable while as independent variables we have KNN filter, ANN filter, random forest ensemble (RFE) model, genetic algorithm (GA), and classifiers as manipulative independent variables. Our experiment follows 1 factor 1 treatment (1F1T) for RQ1 whereas for RQ2, RQ3, and RQ4, there are 1 factor 2 treatments (1F2T) design. We first carried out the explanatory data analysis (EDA) to know the nature of our dataset. Then we pre-processed our data by removing and solving the issues identified. During data preprocessing, we analyze that we have multi-class data; therefore, we first rank features and select multiple feature sets using the info gain algorithm to get maximum variation in features for multi-class dataset. To remove noise, we use ANN-filter and get significant results more than 40% to 60% compared to NN filter with base paper (all, ckloc, IG). Then we applied search-based optimizer i.e., random forest ensemble (RFE) to get the best features set for a software prediction model and we get 30% to 50% significant results compared with genetic instance selection (GIS). Then we used a classifier to predict defects for CPDP. We compare the results of the classifier with base paper classifier using F1-measure and we get almost 35% more than base paper. We validate the experiment using Wilcoxon and Cohen's d test.

Keywords: search-based optimizer; cross project defect prediction; artificial neural network information-gain; ANN filter; K-nearest neighbor (KNN filter); random forest ensemble (RFE)

Citation: Faiz, R.b.; Shaheen, S.; Sharaf, M.; Rauf, H.T. Optimal Feature Selection through Search-Based Optimizer in Cross Project. *Electronics* **2023**, *12*, 514. https://doi.org/10.3390/electronics12030514

Academic Editor: George A. Tsihrintzis

Received: 23 December 2022
Revised: 6 January 2023
Accepted: 9 January 2023
Published: 19 January 2023

1. Introduction

For prediction of software, software defect proneness (SDP) is a study area that provides effective techniques. From previous versions of the same project, defective data can be used to detect fault proneness. At early stages of software development, prediction of defects in software subsystems (modules) plays a vital role in decreasing the development costs and time. It eradicates the excessive efforts to find defects from the software modules in later stages of the software development. Preceding studies in this research area consider the within project defect prediction (WPDP) in which the same data are used for training and predicting defects and are cross-validated [1]. However, according to [2], WPDP approach is only valid when there is a large dataset with less granularity. Yet, such approaches do not hold in training data specifically for inactive software projects.

In order to assure the quality of data, quality assurance activities help software engineers in testing defect modules and in code inspection to consume less time and facilitate more usage. Recently, software defect prediction (SDP) practice specifies defective modules based on metrics of software process and products. Thus, [3,4] use Chidamber and Kemerer (CK) metrics as a feature set to predict software prediction models (SPM) using logistic regression (LR). Whereas [5] used static code, metric (SCM), and object-oriented (OO) metrics. Some researchers used historical data from other projects [6]. Where [7,8] used feature subsets for feature selection i.e., SCM + OO + LOC (all), CK + LOC, and info-gain (IG) subsets.

Furthermore, there are many open-source public datasets available such as Apache, NASA, Tera PROMISE data repositories. However, many researchers provide results to overwhelm the problem of WPDP in context of CPDP. Cross project defect prediction (CPDP) is an approach of using historical data from different projects which has been experimentally proven by different researchers [9,10]. The defects are available on the given available open-source public datasets on the Internet. Therefore, quality of data is very crucial for training data for better software model prediction. Data quality is one of the challenging tasks therefore, for CPDP, there are two solutions to this problem.

One of the solutions given by researchers is feature selection (FS), which is validated on different datasets in order to reduce dimensions in the projects. The results validate that using FS approach improves the performance and efficiency of SDP [1,10]. Second solution used by researchers is reducing the data size of training data [11,12] by selecting suitable and excluding irrelevant training data. Training data selection (TDS) is considered preferable for the simplification of data volume to get better prediction in terms of software defect prediction. Hence, the above-mentioned techniques and methods were used to address the most important issues, such as quality of data and defect-prone modules for software prediction models. Some researchers applied boosting and bagging technique, and others used filter and wrapper approaches and local data from previous projects. Therefore, we used one feature selection (FS) method i.e., info-gain sub setting [7] in conducting research analysis and used all features in our experiment. For filtering approach, we used the ANN-filter; for optimal feature sets, we used the random forest ensemble (RFE) technique; and in the end, we used classifier [13,14] for predicting the software prediction model (SPM).

2. Literature Review

It is evident from the research analysis among all feature selection (FS) methods i.e., Info-Gain (IG) sub setting [7] improved the defect prediction accuracy. Because of which, IG in combination with filter (ANN) for feature selection gives optimal feature for better classification [2]. Besides the existing literature, critical analysis reveals gaps that the dataset is multi-class and has issues in noise, class imbalance, and distribution gap. However, we believe that by combining all these techniques, defect prediction accuracy for multi-class in cross-project can be improved.

Various studies reviewed in the literature have developed software prediction models for estimating the prediction of software product including effort estimation, maintainability, defect proneness, and change proneness during the development lifecycle of the software. These software attributes are used to identify delicate parts of a software product and hence provide good practices to software project managers to examine the delicate parts of the software product. These practices help software project managers to assure the quality of software products.

The authors in [7] proposed an advanced technique in which NN-filter is embedded with genetic instance selection algorithm (GIS) and is applied on 41 Tera Promise Repository of cross project defect prediction (CPDP). To get the optimal solution and to remove noise from the labeling dataset, they applied the GIS approach. For the cleansing of dataset, feature-ranking phenomena is used by using iterative info-gain feature sub setting for feature selection. NN-filter is embedded in the model to create validation set in order to enhance the performance of the cross-project defect prediction (CPDP) by using cross validation. The base paper is defined in context of F1 and G measures which are used as a fitness function (F1*G) in GIS algorithm. The prediction is measured in terms of ROC, AUC, precision, recall, and F-measure. The results show that the performance of the classifier was enhanced by using such approaches. Thus, above research suggest that in case of high dimensional, data filters can improve prediction accuracy if applied before search-based algorithm. Feature Selection (FS) algorithms are proposed in [11,15] i.e., binary genetic algorithm (BGA), binary particle swarm optimization (BPSO), and binary ant colony optimization (BACO) to increase the optimization of software prediction models (SPMs). Nineteen real-world software projects from Tera PROMISE repository are used in this experiment. Layered recurrent neural network (L-RNN) is used as a classifier where feature selection (FS) approach is applied to increase the performance of L-RNN classifier to find software defect prediction problems. The data are split into test and training dataset by converting data into binary form i.e., 0 and 1. Training data are then passed through the classifier and results are cross validated. Results are measured through ROC and AUC metrics and are compared with classification algorithm i.e., artificial neural network (ANN), Naïve Bayes (NB), and decision tree (C4.5). Results show that the feature selection approach has high quality performance as compared with using all fixed features. They [16] explained that software defect prediction (SDP) is used to increase software dependability and test proficiency. Therefore, we need to clarify the features used in the software defect prediction, rather than those features that leads to poor performance of classifiers. However, class imbalance problem in software defect prediction has concerned the software industry and academia projects. They proposed a novel technique to solve class-imbalance using learning method for class imbalance problem both within-project and cross project. They applied stratification-embedded approach with nearest neighbor (STr-NN) to evolve training data with balanced data. They directly employed the stratification technique (STr-NN) for defect prediction in within project [6]. They first mitigate the class distribution between target dataset and source, then applied the STr-NN technique on the distributed data. They conducted experiment on NASA and PROMISE datasets. The implied results have higher precision, recall, and area under curve (AUC).

To optimize software resource allocation, SDP (software defect prediction) helps to optimize testing by generalizing defects before testing defect prone modules. Recent studies explicitly predicted defect-prone modules by using historical data for prediction. Though, one way of prediction of defects is by generalizing predictions from data of other projects. Software defect prediction is examined [17] by selection training data using 34 PROMISE datasets in context of cross project defect prediction (CPDP). By using cross project defect prediction, CPDP results are better by using training data from different projects as compared to the data used from similar projects. They employed five machine learning (ML) algorithms to construct prediction model, including support vector machine (SVM), Naïve Bayes (NB), J48, and decision tree (DT). The predictions are predicted by using recall, precision, and accuracy. Results for recall are greater than 70%, for precision greater than 50%, and for accuracy greater than 75%. As non-defective and defective features have imbalance class, therefore, achieving high precision for defect prediction is hard. Cross project defect prediction (CPDP) is necessary, as local data are used to predict defects. Data filters are required for the improvement of CPDP. The authors in [18] advised filter method for the 44 releases of 14 projects from Tera PROMISE data repository. Support vector machine (SVM) and Naïve Bayes (NB) are used as a classifier. TGF and DCBF are used as a filter technique that improves the results of the classifier. The results are measured through AUC.

The local models are used for defect prediction, and limitation of cross project defect prediction is describe by [1] as the main problem is heterogeneous data in cross project defect prediction (CPDP). They evaluate the pros and cons of cross project defect prediction (CPDP) by evaluating the performance of local models. They investigate both within-defect prediction (WDP) using same data and CPDP by transfer heterogeneous data. The results show that there is a minor difference in comparison between WDP and CPDP. They experimented on PROMISE data and showed that the results of support vector machine (SVM) are better than Naïve Bayes (NB) and C4.5 (decision tree) except random forest (RF) that has similar results as SVM. The results are measured using AUC and F-measure.

In another research [18], the authors proposed a boosting technique to balance the class for cross project defect prediction (CPDP). They applied combine learning algorithm SMOTE along with transfer cost-sensitive boost (TCSBoost) in order to reduce the cost of software prediction model. Experiment is carried out upon PROMISE datasets. They measure the results of Naïve Bayes classifier using F-measure and G-measure.

The authors in [19] presented an approach for software bug prediction in which genetic algorithm is used as a search-based algorithm. The feature is selected using the recombination of feature. The proposed methodology is validated on datasets of NASA and PROMISE repository. The results are compared using machine learning classification algorithms. These classification algorithms are decision tree (DT), Naïve Bayes (NB), and artificial neural network (ANN). The estimation process shows that the proposed approach has high accuracy rate and shows better performance. The proposed approach is measured through software quality attributes precision, recall, F-measure, and MMRE. Another feature selection (FS) technique is formalized in [20] using genetic algorithm and propose MOFES technique. They imply the multi-objective optimization problem and propose an advance technique, MOFES, which is then compared with the baseline approaches such as decision tree (DT), logistic regression (LR). The results show that the technique has better prediction performance by selecting less instances. The computational cost of software is acceptable. The results are cross-validated using k-fold. The AUC metric is used to measure the results.

A combined approach is proposed in [18] using feature selection and ensemble learning algorithm to predict performance of defect prediction. Average probability ensemble (APE) is applied along with SVM classifier to attain the performance of prediction model using NASA datasets that include MC1, PC2, and PC4. They used AUC measure to get the results.

Two multi-purpose approach is described in [9] for cross project defect prediction, first they minimize the misclassified feature cost and maximize the recall using comparing results using different classification algorithms, which includes random forest (RF), decision tree (C45), Naïve Bayes (NB), and logistic regression (LR). Second, they minimize the quality assurance activities from defect-prone projects. They run experiment using 41 multiple versions of Tera PROMISE Repository. Among different ML classification algorithms, logistic regression is highly cost effective than other algorithms which are used as single objective. The results are measured using recall and normalized misclassification cost.

The authors in [11,12,14] conducted experiment on NASA and PROMISE datasets and applied the feature subsets and feature ranking approaches in context of cross project defect prediction CPDP. They used K-nearest neighbor KNN and Naïve Bayes (NB) as a prediction algorithm. For feature selection, they used CFS and best-first search algorithm. They compared the results using AUC measures of all and CSF features sets.

Furthermore, the connectivity-based technique is explored in [21] on 26 public datasets of NASA, PROMISE, and AEEEM in order to overcome the heterogeneity problem of cross project defect prediction CPDP. They explored using five supervised (which includes decision tree, random forest, logistic regression, Naïve Bayes, and J48) and unsupervised algorithms (which includes K-mean clustering, C-mean, neural-gas, and spectral cluster technique). They measured the results using AUC, and found that unsupervised algorithms results are better than supervised algorithms.

The authors in [17] proposed a novel technique for cross projects named as multi transformation (MT+). They used BOX-COX transformation on AEEEM, NASA, and PROMISE datasets to define relevant training sets for cross projects. They compared the results of random forest (RF) with log transformation method. The results are measured by F-measure.

The multi-nomial classification technique is adopted in [22] for cross project defect prediction (CPDP) and within cross project defect prediction (WPDP). They applied random forest and gradient boosting method on PROMISE datasets as an ensemble-learning model. They defined their result using F-measure, AUC, and MAP. They labeled class information as class 0, class 1, and class 2. They identified that proper training data selection is necessary to optimize efficiency of defect prediction.

Authors [23] conducted and evaluated the combination of different filter methods on 16 high-dimensional datasets and used KNN as classifier. They compared the results of different combination of filter method to observe which combination gives the best result with respect to run time and stability aspects.

A novel technique is proposed in [21] for intrusion detection system based on feature selection and clustering algorithm. They used the filter and wrapper method for feature selection based on linear correlation coefficient (FGLCC) algorithm and cuttlefish algorithm (CFA) on KDD Cup 99 large dataset along with decision tree as classifier for proposed method. They achieved high accuracy of 95.03%.

To evaluate the impact of hybrid feature selection on PROMISE repository for cross project defect prediction, the authors proposed feature specific approach [24]. They used random forest and recursive feature elimination as hybrid feature selection along with NN classifier. They evaluated that feature selected through hybrid approach does have higher defect prediction accuracy of AUC 75.98%.

2.1. Gap Analysis

From the above study, we find that many researchers applied different methods including feature selection [7,11], filtering methods [16,17], ensemble techniques [2], classification algorithms [1,21], boosting and search-based optimizer [7,24]; Ref. [25] in order to overcome the heterogeneity of cross-project defect prediction (CPDP) [26]. There are several techniques applied in context of major issues of cross project defect prediction but researchers had not addressed the nature of dataset that how to clean data as public datasets have noise. All techniques are applied but still heterogeneity is the major problem in CPDP. This problem can be overcome by using multi-nominal classification using filters as data contain noise; therefore, we first rank and select relevant feature sets then apply search-based optimizer to optimize the defect prediction.

2.2. Research Questions

To achieve our goal, we focus on following research questions:

- RQ1: What is the impact of feature selection for multi-class compared with binary class on cross-project defect prediction through F1 measure?

  Null Hypothesis: Feature selection for multi-class compared with binary class has no impact on cross-project defect prediction through F1 measure.
  Alternate Hypothesis: Feature selection for multi-class compared with binary class has an impact on cross-project defect prediction through F1 measure.

- RQ2: What is the impact of ANN filter compared with KNN filter on cross-project defect prediction through F1-measure?

  Null Hypothesis: ANN filter compared with KNN filter has no impact on cross-project defect prediction through F1-measure.
  Alternate Hypothesis: ANN-filter compared with KNN-filter has an impact on cross-project defect prediction through F1-measure.

- RQ3: What is the impact of search-based optimizer i.e., random forest ensemble compared with genetic algorithm, on cross-project defect prediction through F1-measure?

  Null Hypothesis: Search-based optimizer i.e., random forest ensemble compared with Genetic Algorithm, has no impact on cross-project defect prediction through F1-measure.

Alternate Hypothesis: Search-based optimizer, i.e., random forest ensemble compared with genetic algorithm, has an impact on cross-project defect prediction through F1-measure.

- RQ4: What is the impact of our classifier compared with Naïve Bayes classifier on cross-project defect prediction through F1-measure?

  Null Hypothesis: Our classifier compared with Naïve Bayes classifier has no impact on cross-project defect prediction through F1-measure.
  Alternate Hypothesis: Our classifier compared with Naïve Bayes classifier has an impact on cross-project defect prediction through F1-measure.

3. Research Methodology

This section defines the thorough facts of our experiment which we carry out. Let us discuss the content in detail.

The context of our research is cross project defect prediction. We predict the defects in cross project using machine learning algorithm. Our research type is explanatory which aims to explain the causes and consequences of a well-defined problem. The cross-project defect prediction is a well-defined problem, and we will find the defect prediction accuracy.

3.1. Data Collection

Our research is based on quantitative research methods to carry out the experiment. As quantitative research methods focus on numbers and statistics, our dataset is also of the same type. Objects of our study are source and target projects of PROMISE repository. In our research, we use different projects of promise repository as input to our model to train our classifier. Our research normalized our datasets before training our classifier (e.g., checking for noise removal, class imbalance, and handling distribution gap between source and target projects).

3.2. Research Method

Our research uses experimental research method to manipulate and control the variables in order to determine the cause and effect between variables for prediction accuracy and authenticity. The purpose of our experiment is optimum feature selection through search-based optimizer for cross project defect prediction. The perspective of our experiment is earlier defect prediction based on already developed mature projects. We will predict the defects earlier by training the classifier using mature source project and then predicting defects in the target projects. Our research has F1 measure as dependent variables. Our research has different independent variables for every research question. For Research question 1, our independent variable is KNN filter. For research question 2, our independent variables are KNN and ANN filter. For third research question, independent variables are random forest ensemble model and genetic algorithm. For last question, we have classifiers as independent variables.

3.3. Research Design

Our experiment design is different for every research question. Let us discuss the design for every question in detail.

- RQ1: What is the impact of feature selection for multi-class compared with binary class on cross-project defect prediction through F1 measure? For this question, design is 1 factor and 1 treatment (1F1T) i.e., KNN-filter.
- RQ2: What is the impact of ANN-filter compared with KNN-filter on cross-project defect prediction through F1-measure? For this question, design is 1 factor and 2 treatment (1F2T) i.e., KNN-filter and ANN-filter.

- RQ3: What is the impact of search-based optimizer i.e., random forest ensemble compared with genetic algorithm, on cross-project defect prediction through F1-measure? For this question, design is 1 factor and 2 treatment (1F2T) i.e., random forest ensemble and genetic algorithm.
- RQ4: What is the impact of our classifier compared with Naïve Bayes classifier on cross-project defect prediction through F1-measure? For this question, design is 1 factor and 2 treatment (1F2T) i.e., our classifier and Naïve Bayes classifier.

4. Proposed Methodology

In this section, we describe the model followed in our research. We conduct our experiment by performing the following visualized experiment (referred Figure 1):

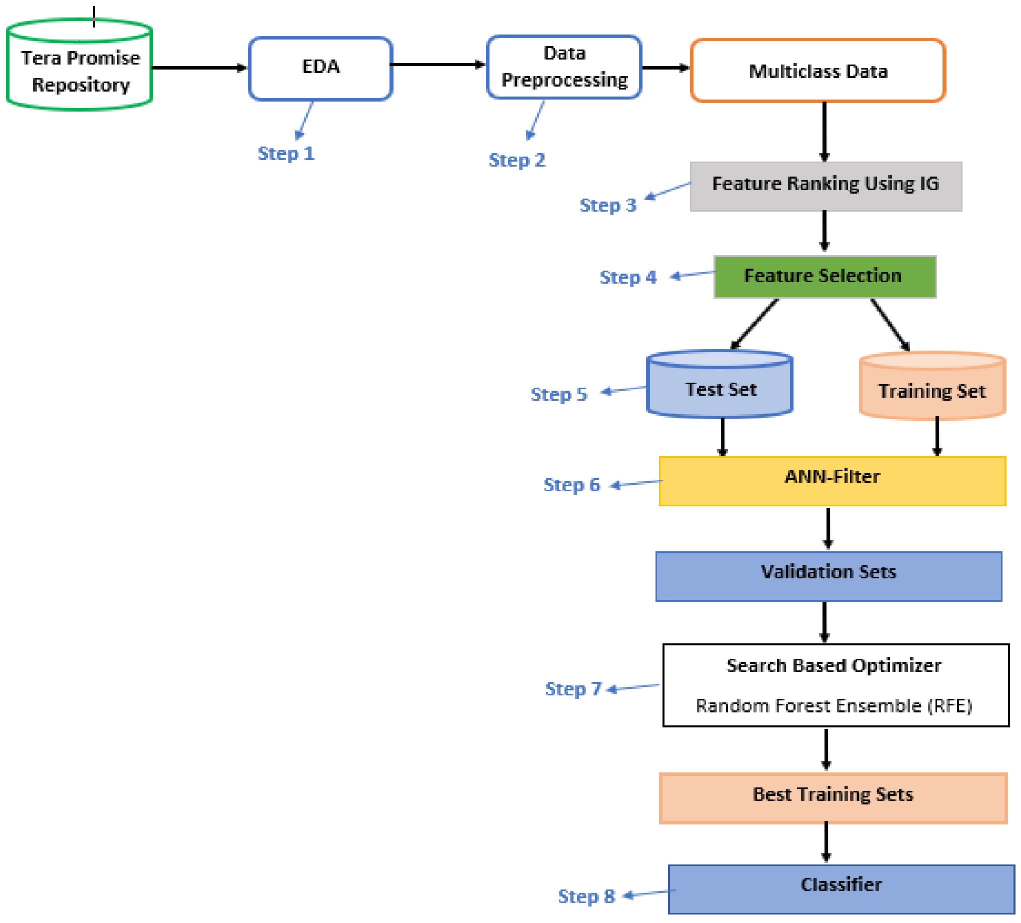


Figure 1. Proposed methodology.

4.1. Promise Repository

For the research, we used the PROMISE repository from year 2018. PROMISE is an open-source repository that is widely used for defect predictions studies. This dataset has 20 sets of features which are as follows (referred Table 1) [1]:

4.2. Exploratory Data Analysis

We explored and analyzed data on PROMISE repository to identify the missing and erroneous data by mapping and analyzing the structure of the data. EDA reveals that PROMISE repository is multi-class as shown in Figure 2 and has noise, class imbalance. The following figures show the variation in data and the presence of multi-class data in multiple versions of the project.

Table 1. Promise repository features.

SR #	Attribute	Abbreviations	Description
1	WMC	Weighted Methods per class	The number of methods used in a given class
2	DIT	Depth of Inheritance Tree	The maximum distance from a given class to the root of an inheritance tree
3	NOC	Number of Children	The number of children of a given class in an inheritance tree
4	CBO	Coupling between Object Classes	The number of classes that are coupled to a given class
5	RFC	Response for a Class	The number of distinct methods invoked by code in a given class
6	LCOM	Lack of Cohesion in Methods	The number of method pairs in a class that do not share access to any class attributes
7	CA	Afferent Coupling	Afferent coupling, which measures the number of classes that depends upon a given class
8	CE	Efferent Coupling	Efferent coupling, which measures the number of classes that a given class depends upon
9	NPM	Number of Public Methods	the number of public methods in a given class
10	LCOM3	Normalized Version of LCOM	Another type of lcom metric proposed by Henderson-Sellers
11	LOC	Lines of Code	The number of lines of code in a given class
12	DAM	Data Access Metric	The ratio of the number of private/protected attributes to the total number of attributes in a given class
13	MOA	Measure of Aggregation	The number of attributes in a given class which are of user-defined types
14	MFA	Measure of Functional Abstraction	The number of methods inherited by a given class divided by the total number of methods that can be accessed by the member methods of the given class
15	CAM	Cohesion among Methods	The ratio of the sum of the number of different parameter types of every method in a given class to the product of the number of methods in the given class and the number of different method parameter types in the whole class
16	IC	Inheritance Coupling	The number of parent classes that a given class is coupled to
17	CBM	Coupling Between Methods	The total number of new or overwritten methods that all inherited methods in a given class are coupled to
18	AMC	Average Method Complexity	The average size of methods in a given class
19	MAX_CC	Maximum Values of Methods in the same Class	The maximum McCabe's cyclomatic complexity (CC) score of methods in a given class
20	AVG_CC	Mean Values of Methods in the same class	The arithmetic means of the McCabe's cyclomatic complexity (CC) scores of methods in a given class

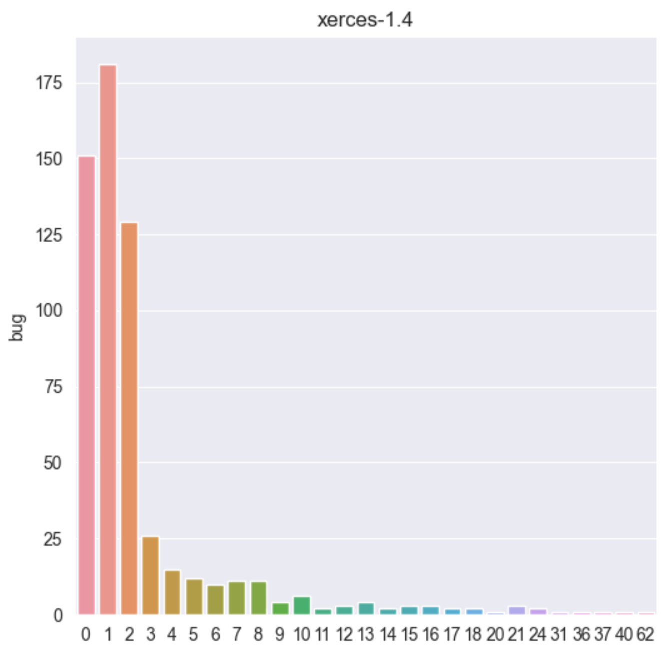
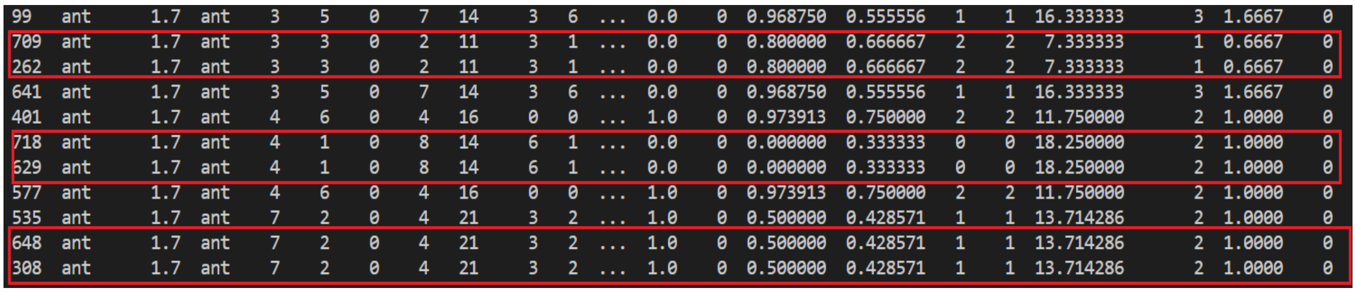


Figure 2. Multi-class of Xerces-1.4.

Our repository has noise in term of duplicated rows (referred to Figure 3) which are 709, 262, 718, 629, 648, and 308. We removed the noise from our PROMISE dataset, so that our trained model predicts the defects accurately with higher accuracy.



```
99   ant    1.7  ant   3   5   0   7  14   3  6  ...  0.0   0  0.968750  0.555556  1   1  16.333333   3  1.6667  0
709  ant    1.7  ant   3   3   0   2  11   3  1  ...  0.0   0  0.800000  0.666667  2   2   7.333333   1  0.6667  0
262  ant    1.7  ant   3   3   0   2  11   3  1  ...  0.0   0  0.800000  0.666667  2   2   7.333333   1  0.6667  0
641  ant    1.7  ant   3   5   0   7  14   3  6  ...  0.0   0  0.968750  0.555556  1   1  16.333333   3  1.6667  0
401  ant    1.7  ant   4   6   0   4  16   0  0  ...  1.0   0  0.973913  0.750000  2   2  11.750000   2  1.0000  0
718  ant    1.7  ant   4   1   0   8  14   6  1  ...  0.0   0  0.000000  0.333333  0   0  18.250000   2  1.0000  0
629  ant    1.7  ant   4   1   0   8  14   6  1  ...  0.0   0  0.000000  0.333333  0   0  18.250000   2  1.0000  0
577  ant    1.7  ant   4   6   0   4  16   0  0  ...  1.0   0  0.973913  0.750000  2   2  11.750000   2  1.0000  0
535  ant    1.7  ant   7   2   0   4  21   3  2  ...  1.0   0  0.500000  0.428571  1   1  13.714286   2  1.0000  0
648  ant    1.7  ant   7   2   0   4  21   3  2  ...  1.0   0  0.500000  0.428571  1   1  13.714286   2  1.0000  0
308  ant    1.7  ant   7   2   0   4  21   3  2  ...  1.0   0  0.500000  0.428571  1   1  13.714286   2  1.0000  0
```

Figure 3. Noise as duplicated rows in data.

Our repository has an issue of distribution gap between source and target projects which diversely affect the prediction accuracy of the classifier. So, we removed the outliers from the datasets. The visualization for DIT feature of Camel and Ant is given in Figures 4 and 5 respectively.

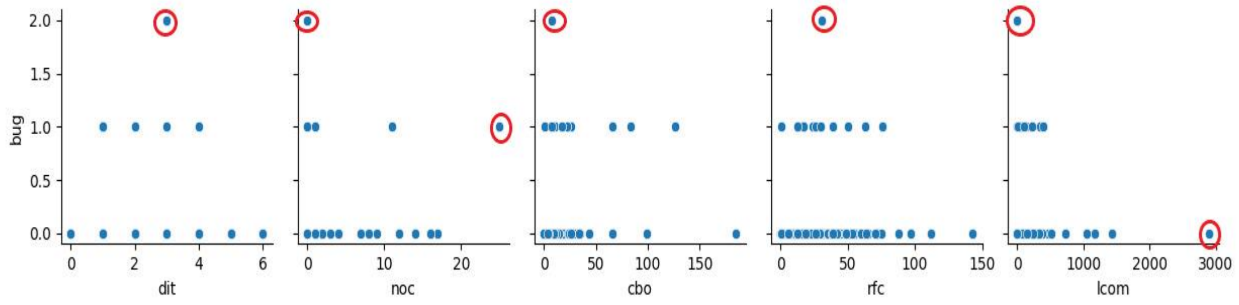


Figure 4. DIT feature of Camel. Dots circled in red represents the outliers.

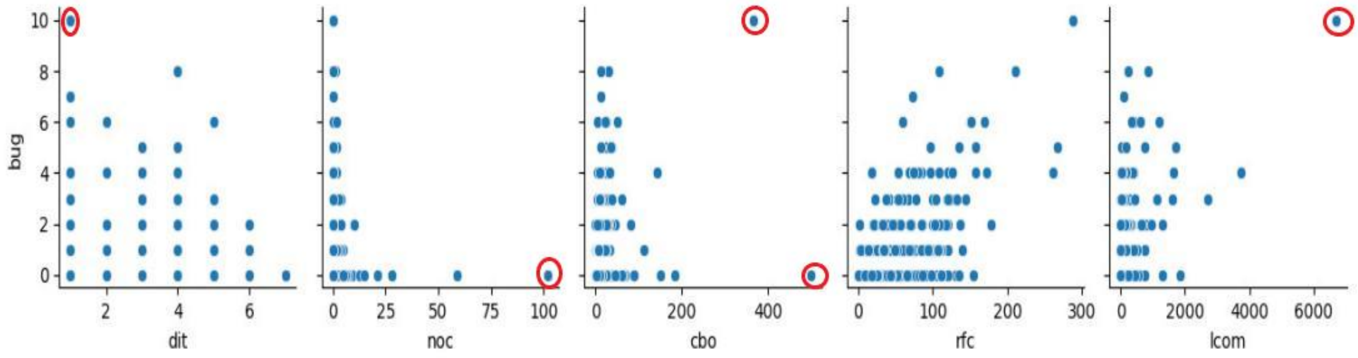


Figure 5. DIT feature of Ant. Dots circled in red represents the outliers.

4.3. Data Preprocessing

In this section, we applied different techniques in order to normalize our data for further experimentation. Normalizing data means to remove all the issues which we identified in EDA of our repository. Our data contain numeric values in which different numbers represent the whole class data. The classes contain (0–62). Therefore, it is required to sort data so that we can distinguish the total number of classes. After sorting data, we clean the data, which leads to over-fitting of the model. Therefore, we remove the outliers which contain the data that are not required for training. Our dataset is unbalanced, which means it contains more samples for some classes compared to other datasets. Therefore, we balance our dataset before training our classifier in order to get equal number of samples in all our datasets. For balancing data, we use feature sampling.

4.4. Feature Ranking

Feature ranking is the procedure of selecting subset of significant instances in building software model. The significance of feature ranking is the generalization of models to make them easier to investigate by researchers and to avoid high dimensional data. By reducing over-fitting, feature ranking improves generalization. In our research, we use info-gain, a measure of the feature sub setting, which explains highest entropy in the data. The info-gain is formulated as:

$$HC = -c\epsilon Cpcp(c) \tag{1}$$

$$HA = -c\epsilon Ap(a)c\epsilon Cpap(c \mid a) \tag{2}$$

Using the info-gain approach, the features of the datasets are ranked from the highest to the lowest amount of entropy explained. The iterative Info-Gain is used to select subsets of applicable set of instances, computed by using top ranked attributes of training set. The predictor (j + 1) ranks these attributes continuously until all attributes are ranked for predictions. After feature ranking, we select the features and remove the redundant data from the feature subsets using info gain.

4.5. Feature Selection

We train and test three different sets of features for the base paper, which includes CK + LOC, the whole sets of features contain OO + SCM + LOC (all) and the third set contains IG (Info Gain) features which are extracted through iterative Info Gain sub setting technique using entropy. For CK + LOC, the base paper included features such as wmc, dit, noc, cbo, rfc, lcom, loc. For OO + SCM + LOC (all), base paper contains whole features that are defined in datasets. While IG features have, different sets of features for different versions of project that are given in Table 2.

Table 2. Info gain features of datasets.

SR #	Datasets	Info Gain Features
1	Ant-1.3	rfc, ca, lcom3, cam, avg_cc
2	Ant-1.4	ce, loc, moa
3	Ant-1.5	wmc, cbo, rfc, ce, cbm
4	Ant-1.6	rfc, lcom3, cam, cbm, amc, avg_cc
5	Ant-1.7	cbo, ce, loc, cbm, amc, avg_cc
6	Camel-1.0	cbo, lcom, ca, npm, mfa
7	Camel-1.2	ca, ce, npm, mfa, cbm
8	Camel-1.4	rfc, ce, lcom3, loc, mfa, avg_cc
9	Camel-1.6	dit, lcom, ca, ce, amc, max_cc
10	Ivy-1.1	lcom, ca, amc, max_cc
11	Ivy-1.4	ca, ce, lcom3, loc, cam
12	Ivy-2.0	cbo, lcom, npm, ic, max_cc
13	Jedit-3.5	cbo, rfc, ca, ce, moa, max_cc
14	Jedit-4.0	lcom, ca, locm3, loc, moa, cam
15	Jedit-4.1	rfc, loc, amc, avg_cc
16	Jedit-4.2	rfc, npm, dam, mfa
17	Jedit-4.3	ca, ce, npm, loc, moa
18	Log4j-1.0	lcom, npm, loc, moa
19	Log4j-1.1	wmc, cbo, loc, mfa
20	Log4j-1.2	rfc, ca, npm, mfa, ic, cbm, amc
21	Lucene-2.0	noc, rfc, moa, mfa, cam, amc
22	Lucene-2.2	ca, npm, lcom3, loc, moa, amc
23	Lucene-2.4	rfc, ca, ce, lcom3, dam, amc
24	Poi-1.5	cbo, rfc, lcom3, loc, cam
25	Poi-2.0	wmc, cbo, lcom, loc, mfa, amc, max_cc
26	Poi-2.5	loc, dam, cam, cbm, amc
27	Poi-3.0	cbo, ce, lcom3, cbm, amc, avg_cc

Table 2. *Cont.*

SR #	Datasets	Info Gain Features
28	Synapse-1.0	dit, rfc, lcom3, mfa, cam
29	Synapse-1.1	cbo, rfc, ca, npm, mfa, avg_cc
30	Synapse-1.2	rfc, lcom, loc, cbm, amc, avg_cc
31	Velocity-1.4	dit, ce, cam, amc, max_cc
32	Velocity-1.5	noc, lcom, loc, mfa, cam
33	Velocity-1.6	cbo, lcom3, mfa, cam, amc, avg_cc
34	Xalan-2.4	cbo, rfc, ca, loc, amc
35	Xalan-2.5	lcom, loc, cam, cbm
36	Xalan-2.6	rfc, loc, mfa, amc, max_cc
37	Xalan-2.7	rfc, loc, mfa, amc, max_cc
38	Xerces-1.2	noc, cbo, rfc, npm, moa, cam
39	Xerces-1.3	cbo, ca, loc, dam, moa
40	Xerces-1.4	cbo, ca, ce, loc, mfa, avg_cc
41	Xerces-init	wmc, cbo, loc, dam, moa, amc, avg_cc

4.6. Search Based Optimizer

The data are split into source and target datasets. The classifier is trained using source project and tested on the target project. After splitting the datasets into source and target projects, the source projects are fed into ANN-filter. The projects are fed into this filter in order to get the best possible feature sets. On getting the best sets of features, we give the sets to search-based optimizer i.e., random forest ensemble model and genetic algorithm as well for multi-class feature selection. We compare the results of both the techniques.

In the last step, we classify our results into multi-class by applying the trained model on the target projects and compared our results with the base paper classifier. In this section, we explain results of our all-research questions one by one and then we analyze them in detail.

RQ1: What is the impact of feature selection for multi-class compared with binary class on cross project defect prediction through F1 measure?

We used 41 datasets of Tera PROMISE Repository for our research experiment. The datasets have multi-class problem but most of the research are carried out on these datasets by using binary class. Therefore, we analyzed and compared the results by carrying out experiments for both binary and multi-class through KNN and then we compared our results with the base paper. Through results given in Table 3, we describe and explain the reasons to use multi-class data for the research experiment.

By comparing the multi-class and binary class results with the base paper OO + SCM + LOC (all), CKLOC, IG, in Table 3, it is seemed that the multi-class F1-measures are significantly better than binary class F1-measures. The dataset Xerces-1.3 has high F1-measure, which is 0.92 for base paper OO + SCM + LOC (all) for multi-class-KNN, whereas for binary-KNN has a value of 0.33. Similarly, Ivy-2.0 has more F1-score for multi-class-KNN (all) than binary-KNN (all) which has 0.36 F1-score. These results are present in Table 3. Whereas Lucene-2.4 has the lowest value, 0.29 for multi-class-KNN as compared to binary-KNN, which has 0.35. Overall results are significantly improved by 40% to 60% using multi-class for KNN algorithm for the base paper set for the research experiment.

RQ2: What is the impact of ANN filter compared with KNN filter on cross project defect prediction through F1-measure?

Datasets are imported using sklearn library in python after preprocessing of data, and the data are fed into ANN-filter. Using ANN (artificial neural network) algorithm, we filter out the data to select the most relevant features. ANN is used in different ways such as MLP (multi-layer perceptron), SLP (single layer perceptron), and RBF (radial basis function). For our experiment, we use MLP, as we have multi-class problem. Therefore, we compare our results with the base paper shown below in Table 4.

Table 3. Multi-class vs binary class of dataset.

		Multi-Class			**Binary-Class**		
		ANN (Artificial Neural Network) Filter			**KNN (K-Nearest Neighbor) Filter**		
SR #	**Datasets**	**ALL**	**CKLOC**	**IG**	**ALL**	**CKLOC**	**IG**
1	Ant-1.3	0.88	0.89	0.87	0.37	0.44	0.48
2	Ant-1.4	0.66	0.68	0.70	0.21	0.18	0.27
3	Ant-1.5	0.87	0.85	0.86	0.31	0.44	0.33
4	Ant-1.6	0.67	0.71	0.67	0.41	0.42	0.45
5	Ant-1.7	0.75	0.76	0.75	0.49	0.42	0.48
6	Camel-1.0	0.74	0.66	0.94	0.18	0.23	0.12
7	Camel-1.2	0.59	0.58	0.58	0.27	0.23	0.24
8	Camel-1.4	0.81	0.83	0.82	0.28	0.23	0.26
9	Camel-1.6	0.79	0.78	0.79	0.21	0.21	0.22
10	Ivy-1.1	0.60	0.65	0.43	0.37	0.22	0.22
11	Ivy-1.4	0.89	0.88	0.88	0.31	0.12	0.18
12	Ivy-2.0	0.91	0.88	0.88	0.36	0.43	0.41
13	Jedit-3.5	0.78	0.75	0.68	0.33	0.22	0.30
14	Jedit-4.0	0.79	0.80	0.79	0.42	0.30	0.36
15	Jedit-4.1	0.83	0.87	0.81	0.49	0.41	0.40
16	Jedit-4.2	0.89	0.89	0.87	0.44	0.37	0.42
17	Jedit-4.3	0.67	0.72	0.97	0.09	0.16	0.15
18	Log4j-1.0	0.59	0.60	0.67	0.51	0.39	0.34
19	Log4j-1.1	0.55	0.58	0.55	0.57	0.50	0.46
20	Log4j-1.2	0.35	0.43	0.42	0.21	0.13	0.16
21	Lucene-2.0	0.50	0.46	0.51	0.44	0.32	0.38
22	Lucene-2.2	0.38	0.36	0.36	0.28	0.22	0.23
23	Lucene-2.4	0.29	0.37	0.38	0.35	0.21	0.25
24	Poi-1.5	0.75	0.78	0.68	0.31	0.21	0.21
25	Poi-2.0	0.85	0.84	0.85	0.26	0.19	0.23
26	Poi-2.5	0.65	0.64	0.53	0.23	0.16	0.17
27	Poi-3.0	0.66	0.63	0.64	0.26	0.18	0.19
28	Synapse-1.0	0.78	0.79	0.84	0.42	0.31	0.41
29	Synapse-1.1	0.63	0.64	0.63	0.46	0.31	0.44
30	Synapse-1.2	0.57	0.55	0.57	0.56	0.31	0.43
31	Velocity-1.4	0.49	0.46	0.47	0.18	0.08	0.13
32	Velocity-1.5	0.54	0.51	0.51	0.22	0.11	0.18
33	Velocity-1.6	0.60	0.63	0.66	0.29	0.20	0.31
34	Xalan-2.4	0.85	0.85	0.84	0.39	0.31	0.34
35	Xalan-2.5	0.61	0.56	0.58	0.37	0.30	0.300
36	Xalan-2.6	0.67	0.62	0.61	0.51	0.40	0.41
37	Xalan-2.7	0.82	0.81	0.81	0.40	0.24	0.25
38	Xerces-1.2	0.80	0.82	0.79	0.24	0.17	0.20
39	Xerces-1.3	0.92	0.93	0.91	0.33	0.29	0.28
40	Xerces-1.4	0.74	0.73	0.68	0.31	0.18	0.19
41	Xerces-init	0.55	0.56	0.55	0.31	0.25	0.27
	Mean	0.689268	0.691098	0.69122	0.344341	0.273024	0.29778
	Median	0.67	0.71	0.68	0.331	0.239	0.277

There is a combination of neurons (more linear layers) in MLP (multi-layer perceptron). Usually there are three layers in the three-layer network of neuron. Input layer is the first layer, hidden layer is the middle layer, and output layer is the last layer. We feed data into input layer and get output from output layer. The number of hidden layers can increase as much as it is required for training data.

To train MLP model, we use solver "ibfgs" where two hidden layers are used in which 50 and 5 weights are assign to them. We give maximum 1000 iterations to the model where random state = 0. The results of this model are shown in Table 4. The following Table 4 shows the results of ANN filter and KNN filter for multi-class. The results are compared with the base paper OO + SCM + LOC (all), CK + LOC (ckloc), and Info Gain (IG) using F1-score.

Table 4. ANN vs. KNN result of datasets.

		Multi-Class					
SR #	Datasets	ANN (Artificial Neural Network) Filter			KNN (K-Nearest Neighbor) Filter		
		ALL	CKLOC	IG	ALL	CKLOC	IG
1	Ant-1.3	0.95	0.57	0.50	0.88	0.89	0.87
2	Ant-1.4	0.94	0.85	0.33	0.66	0.68	0.70
3	Ant-1.5	0.93	0.90	0.75	0.87	0.85	0.86
4	Ant-1.6	0.92	0.94	0.72	0.67	0.71	0.67
5	Ant-1.7	0.99	0.97	0.92	0.75	0.76	0.75
6	Camel-1.0	0.93	0.96	0.85	0.74	0.66	0.94
7	Camel-1.2	0.97	0.92	0.53	0.59	0.58	0.58
8	Camel-1.4	0.97	0.99	0.68	0.81	0.83	0.82
9	Camel-1.6	0.97	0.98	0.83	0.79	0.78	0.79
10	Ivy-1.1	0.90	0.83	0.71	0.60	0.65	0.43
11	Ivy-1.4	0.93	0.95	0.92	0.89	0.88	0.88
12	Ivy-2.0	0.97	0.98	0.92	0.91	0.88	0.88
13	Jedit-3.5	0.88	0.96	0.49	0.78	0.75	0.68
14	Jedit-4.0	0.86	0.91	0.88	0.79	0.80	0.79
15	Jedit-4.1	0.90	0.96	0.90	0.83	0.87	0.81
16	Jedit-4.2	0.96	0.96	0.57	0.89	0.89	0.87
17	Jedit-4.3	0.97	0.98	0.87	0.67	0.72	0.97
18	Log4j-1.0	0.92	0.96	0.62	0.59	0.60	0.67
19	Log4j-1.1	0.90	0.95	0.57	0.55	0.58	0.55
20	Log4j-1.2	0.85	0.95	0.33	0.35	0.43	0.42
21	Lucene-2.0	0.92	0.90	0.18	0.50	0.46	0.51
22	Lucene-2.2	0.93	0.96	0.50	0.38	0.36	0.36
23	Lucene-2.4	0.96	0.98	0.59	0.29	0.37	0.38
24	Poi-1.5	0.96	0.96	0.23	0.75	0.78	0.68
25	Poi-2.0	0.97	0.96	0.85	0.85	0.84	0.85
26	Poi-2.5	0.95	0.95	0.31	0.65	0.64	0.53
27	Poi-3.0	0.91	0.93	0.95	0.66	0.63	0.64
28	Synapse-1.0	0.89	0.96	0.97	0.78	0.79	0.84
29	Synapse-1.1	0.90	0.95	0.53	0.63	0.64	0.63
30	Synapse-1.2	0.98	0.98	0.54	0.57	0.55	0.57
31	Velocity-1.4	0.97	0.96	0.71	0.49	0.46	0.47
32	Velocity-1.5	0.96	0.96	0.33	0.54	0.51	0.50
33	Velocity-1.6	0.97	0.87	0.50	0.60	0.63	0.66
34	Xalan-2.4	0.93	0.98	0.85	0.85	0.85	0.84
35	Xalan-2.5	0.98	0.99	0.88	0.61	0.56	0.58
36	Xalan-2.6	0.97	0.98	0.90	0.67	0.62	0.61
37	Xalan-2.7	0.98	0.98	0.80	0.82	0.81	0.81
38	Xerces-1.2	0.98	0.89	0.80	0.80	0.82	0.79
39	Xerces-1.3	0.94	0.97	0.89	0.92	0.93	0.91
40	Xerces-1.4	0.99	0.97	0.94	0.74	0.73	0.68
41	Xerces-init	0.87	0.94	0.58	0.55	0.56	0.55
	Mean	0.943732	0.942707	0.678537	0.689268	0.691098	0.69122
	Median	0.95	0.96	0.714	0.67	0.71	0.68

Xerces-1.4 has higher F1-score (99.1%) for the base paper (all) for ANN-filter, whereas Xerces 1.4 has (74%) F1-score for KNN filter. Log4j 1.2 has least F1-score (85.7%) for the base paper (all) for ANN-filter while Log4j-1.2 has (35%) F1-score for KNN-filter. The results are improved more than 45%.

The results of ANN filter (ckloc) and KNN filter (ckloc) show that the results of projects including camel-1.4, xalan-2.5, xalan-2.4, Jedit 4.3, and ivy-2.0 have high F1-measure for ANN filter (0.992, 0.990, 0.989, 0.988, and 0.986) compared to KNN filter (0.83, 0.56, 0.85, 0.72, and 0.88) respectively. Where ant 1.3 has the lowest F1-measure for ANN filter of 0.571 compare with KNN filter which is 0.897 using multi-class.

The results of ANN filter (IG) and KNN filter (IG) show that the results of projects including synapse-1.0, poi-3.0, xerces-1.4, ant-1.7, and ivy 1.4 have high F1-measure for ANN filter (0.97, 0.952, 0.947, 0.923, and 0.920) comparing to KNN filter (0.84, 0.64, 0.68, 0.75, and 0.88) respectively. Where lucene-2.0 has the lowest F1-measure for ANN filter of 0.18 compare with KNN filter, which is 0.51 using multi-class IG feature sets.

RQ3: What is the impact of search-based optimizer i.e., random forest ensemble compared with base paper genetic algorithm, on cross project defect prediction through F1-measure?

We applied the random forest ensemble as a search-based optimizer for multi-class feature sets. The results are shown in Table 5; we compared the results of search-based optimizer random forest ensemble (RFE) with genetic algorithm (GA) using base paper SCM + OO + LOC (all), CK + LOC (ckloc), and IG features.

From Table 5, it is shown the results random forest ensemble (all) including projects (poi-2.5, poi-3.0, xerces-1.4, and ant-1.5) have high F1-measure which is 0.99, 0.98, 0.980, and 0.97 respectively comparing with genetic algorithm which has values of 0.769, 0.767, 0.665, 0.13 respectively. Where log4j-1.2 has the lowest F1-measure for RFE (all) of 0.67 compare with GA (all), which is 0.746 using multi-class IG feature sets.

Table 5. RFE vs. GA result of datasets.

SR #	Datasets	Random Forest Ensemble (RFE)			Genetic Algorithm (GA)		
		ALL	CKLOC	IG	ALL	CKLOC	IG
1	Ant-1.3	0.88	0.80	0.54	0.38	0.43	0.41
2	Ant-1.4	0.88	0.78	0.59	0.44	0.39	0.41
3	Ant-1.5	0.97	0.60	0.72	0.31	0.35	0.35
4	Ant-1.6	0.92	0.86	0.69	0.50	0.52	0.55
5	Ant-1.7	0.96	0.90	0.90	0.45	0.48	0.50
6	Camel-1.0	0.96	0.93	0.58	0.20	0.19	0.20
7	Camel-1.2	0.96	0.90	0.78	0.52	0.58	0.51
8	Camel-1.4	0.91	0.93	0.57	0.39	0.41	0.38
9	Camel-1.6	0.95	0.95	0.52	0.40	0.44	0.38
10	Ivy-1.1	0.84	0.60	0.84	0.66	0.70	0.60
11	Ivy-1.4	0.94	0.93	0.78	0.24	0.27	0.27
12	Ivy-2.0	0.95	0.93	0.53	0.36	0.31	0.41
13	Jedit-3.5	0.87	0.90	0.18	0.56	0.61	0.60
14	Jedit-4.0	0.89	0.88	0.65	0.46	0.50	0.51
15	Jedit-4.1	0.89	0.83	0.53	0.52	0.52	0.56
16	Jedit-4.2	0.92	0.85	0.73	0.38	0.36	0.43
17	Jedit-4.3	0.89	0.88	0.84	0.11	0.09	0.12
18	Log4j-1.0	0.85	0.84	0.85	0.49	0.56	0.51
19	Log4j-1.1	0.90	0.95	0.85	0.58	0.63	0.61
20	Log4j-1.2	0.67	0.93	0.48	0.74	0.69	0.79
21	Lucene-2.0	0.91	0.97	0.29	0.63	0.65	0.61
22	Lucene-2.2	0.96	0.98	0.69	0.64	0.68	0.61
23	Lucene-2.4	0.96	0.97	0.76	0.69	0.71	0.66
24	Poi-1.5	0.88	0.88	0.40	0.68	0.70	0.74
25	Poi-2.0	0.93	0.97	0.80	0.28	0.31	0.32
26	Poi-2.5	0.99	0.98	0.55	0.76	0.76	0.80
27	Poi-3.0	0.98	0.98	0.90	0.76	0.80	0.79
28	Synapse-1.0	0.95	0.91	0.97	0.29	0.33	0.34
29	Synapse-1.1	0.94	0.96	0.60	0.46	0.52	0.51
30	Synapse-1.2	0.97	0.98	0.58	0.55	0.57	0.57
31	Velocity-1.4	0.92	0.91	0.75	0.57	0.64	0.72
32	Velocity-1.5	0.95	0.94	0.64	0.63	0.60	0.71
33	Velocity-1.6	0.90	0.94	0.27	0.51	0.53	0.56
34	Xalan-2.4	0.91	0.95	0.84	0.39	0.38	0.40
35	Xalan-2.5	0.92	0.92	0.20	0.57	0.59	0.58
36	Xalan-2.6	0.89	0.92	0.31	0.52	0.58	0.59
37	Xalan-2.7	0.93	0.95	0.31	0.81	0.84	0.78
38	Xerces-1.2	0.92	0.94	0.77	0.24	0.27	0.28
39	Xerces-1.3	0.97	0.97	0.88	0.42	0.35	0.40
40	Xerces-1.4	0.98	0.93	0.84	0.66	0.65	0.71
41	Xerces-init	0.89	0.89	0.52	0.40	0.43	0.51
	Mean	0.918293	0.902780	0.634634	0.495878	0.515	0.524073
	Median	0.92	0.93	0.65	0.507	0.529	0.519

It is shown that the results of random forest ensemble (ckloc) including projects (lucene-2.2, poi-2.5, poi-3.0, xerces-1.4, and ant-1.5) have high F1-measure which is 0.99, 0.98, 0.980, and 0.97 respectively comparing with genetic algorithm which has values (0.769, 0.767, 0.665, 0.13 respectively). Where log4j-1.2 has the lowest F1-measure for RFE (all) of 0.67 compared with GA (all), which is 0.746 using multi-class IG feature sets.

The results of random forest ensemble (IG) including projects (lucene-2.2, poi-2.5, poi-3.0, xerces-1.4, and ant-1.5) show high F1-measure which is 0.99, 0.98, 0.980, and 0.97 respectively, comparing with genetic algorithm which has values 0.769, 0.767, 0.665, 0.13 respectively. Where log4j-1.2 has the lowest F1-measure for RFE (IG) that has 0.67 compare with GA (IG), which is 0.746 using multi-class IG feature sets.

The results of multi-class random forest ensemble (RFE) and genetic algorithm (GA) for IG features, which are significantly, improve 30% to 50%. In addition, results are also compromised using random forest ensemble (RFE) including ant-1.3, Poi-1.5, xalan-2.7, velocity-1.6, Xerces-init, and jedit-3.5.

RQ4: What is the impact of our classifier compared with Naïve Bayes classifier on cross project defect prediction through F1-measure?

In this approach, the complete training set fed into the learner and the model is trained with all the training data points. For the prediction measure, we use a classifier. Table 6 illustrates the classifier by comparing with Naïve Bayes benchmark classifier SCM + OO + LOC (all), CK + LOC (ckloc), and IG using F1-measure.

From Table 6, it is shown that the results of classifier (all) including projects (lucene-2.2, poi-2.5, poi-3.0, xerces-1.4, and ant-1.5) have Naïve Bayes algorithm that has values (0.769, 0.767, 0.665, and 0.13 respectively). Where log4j-1.2 has the lowest F1-measure for RFE (all) of 0.67 compare with GA (all), which is 0.746 using multi-class IG feature sets.

The results of classifier (ckloc) including projects (lucene-2.2, poi-2.5, poi-3.0, xerces-1.4, and ant-1.5) have Naïve Bayes algorithm that has values (0.769, 0.767, 0.665, and 0.13 respectively). Where log4j-1.2 has the lowest F1-measure for RFE (all) of 0.67 compared with GA (all), which is 0.746 using multi-class ckloc feature sets.

Some results are substantially improved and some results are compromised using IG features for software defect prediction. These projects include Jedit-3.5, Xalan-init, Xalan-2.6, Lucene 2.0, Ant-1.4, and Xalan-2.5 which have lower values compared with Naïve Bayes using IG features. The mean values of F1-measure were derived for the experiment using algorithms. From the experiment, we measure F1 to predict the performance of algorithms. The error bar shows the linearity between the results. The results measured for the experiment are shown by using Box Plot as shown in Figure 6.

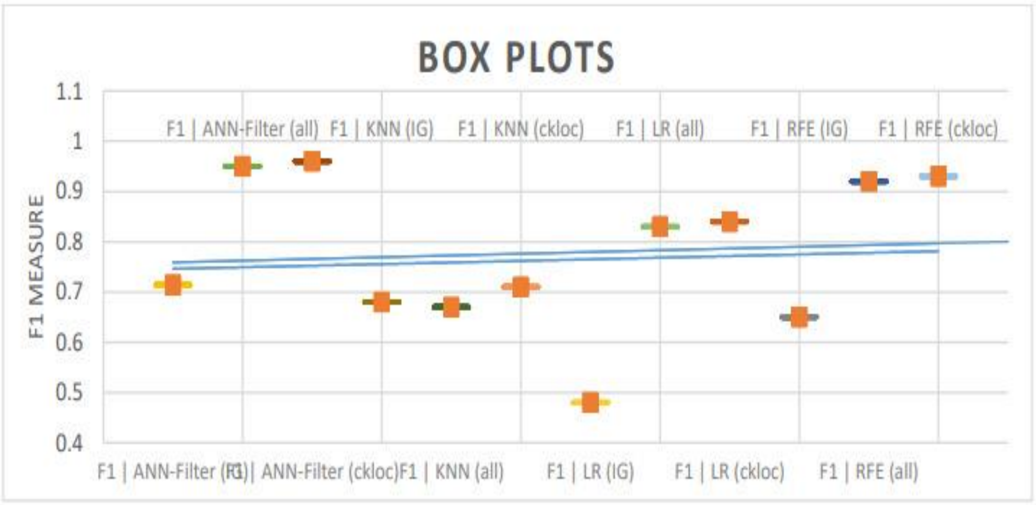


Figure 6. Box plot of F-measure for algorithm used for CPDP.

Table 6. Result of our classifier.

SR #	Datasets	Our Classifier			Benchmark Classifier (NB)		
		ALL	CKLOC	IG	ALL	CKLOC	IG
1	Ant-1.3	**0.70**	0.60	0.51	**0.29**	0.25	0.38
2	Ant-1.4	0.66	**0.71**	0.13	0.19	**0.13**	0.20
3	Ant-1.5	0.80	0.66	0.31	0.34	0.41	0.42
4	Ant-1.6	0.78	0.75	0.48	0.42	0.43	0.46
5	Ant-1.7	0.91	0.90	0.66	0.47	0.45	0.52
6	Camel-1.0	0.83	0.88	0.56	0.34	0.34	0.19
7	Camel-1.2	0.88	0.79	0.57	0.25	0.25	0.24
8	Camel-1.4	0.88	0.86	0.55	0.21	0.22	0.26
9	Camel-1.6	0.92	0.87	0.46	0.20	0.26	0.23
10	Ivy-1.1	0.65	0.52	0.65	0.39	0.35	0.34
11	Ivy-1.4	0.74	0.75	0.48	0.30	0.28	0.30
12	Ivy-2.0	0.81	0.85	0.52	0.39	0.39	0.42
13	Jedit-3.5	0.84	0.80	0.20	0.50	0.39	0.45
14	Jedit-4.0	0.71	0.76	0.38	0.46	0.47	0.51
15	Jedit-4.1	0.78	0.84	0.41	0.60	0.53	0.57
16	Jedit-4.2	0.76	0.83	0.38	0.48	0.47	0.48
17	Jedit-4.3	0.75	0.80	0.49	0.14	0.17	0.16
18	Log4j-1.0	0.84	0.79	0.43	0.38	0.29	0.24
19	Log4j-1.1	0.79	0.78	0.34	0.33	0.16	0.28
20	Log4j-1.2	0.60	0.84	0.39	0.25	0.19	0.19
21	Lucene-2.0	0.69	0.97	0.14	0.27	0.27	0.33
22	Lucene-2.2	0.86	0.88	0.39	0.29	0.23	0.23
23	Lucene-2.4	0.94	0.97	0.78	0.37	0.34	0.31
24	Poi-1.5	0.88	0.88	0.25	0.38	0.30	0.33
25	Poi-2.0	0.88	0.97	0.28	0.23	0.21	0.25
26	Poi-2.5	0.91	0.93	0.48	0.35	0.26	0.34
27	Poi-3.0	0.92	0.94	0.79	0.36	0.30	0.39
28	Synapse-1.0	0.65	0.75	0.85	0.33	0.27	0.33
29	Synapse-1.1	0.82	0.87	0.35	0.38	0.29	0.30
30	Synapse-1.2	0.96	0.99	0.47	0.45	0.32	0.33
31	Velocity-1.4	0.79	0.80	0.48	0.18	0.17	0.21
32	Velocity-1.5	0.92	0.93	0.63	0.26	0.20	0.30
33	Velocity-1.6	0.77	0.88	0.66	0.32	0.32	0.34
34	Xalan-2.4	0.81	0.88	0.83	0.38	0.32	0.40
35	Xalan-2.5	0.90	0.92	0.09	0.41	0.33	0.34
36	Xalan-2.6	0.86	0.88	0.16	0.50	0.44	0.44
37	Xalan-2.7	0.87	0.95	0.17	0.51	0.38	0.38
38	Xerces-1.2	0.88	0.80	0.63	0.24	0.20	0.24
39	Xerces-1.3	0.83	0.82	0.74	0.33	0.33	0.29
40	Xerces-1.4	0.88	0.80	0.82	0.37	0.30	0.30
41	Xerces-init	0.79	0.70	0.23	0.35	0.37	0.36
	Mean	0.81561	0.83146341	0.466341	0.350659	0.31204878	0.334976
	Median	0.83	0.84	0.48	0.35	0.303	0.331

The above box-plot clearly explains the results of all the algorithms used in our experiment i.e., ANN filter, classifiers, and feature selection technique through F1 measure. The figure illustrates the mean values of F1-measure conducted for the experiment using algorithms. From the experiment, we measure the F1 to predict the performance of algorithms. The error bar shows the linearity between the results.

5. Research Validation

In this section, we describe the validity of our research through statistical tests i.e., Cohen's D, Glass's Delta and Hedges' G, and Wilcoxon test. Let us review the results in detail.

5.1. Cohen's D

Cohen's d is calculated for the individual T test samples by measuring the mean variance of two sets and then dividing the estimate by the combined standard deviation.

$$\text{Cohen's d} = (\text{M2} - \text{M1})/\text{SDpooled} \tag{3}$$

where:

$$\text{SDpooled} = \mu\ ((\text{SD12} + \text{SD22})/2) \tag{4}$$

5.2. Glass's Delta and Hedges' G

Cohen's d is an effective of effective size if two classes have similar standard deviations and are of the equal size. Glass's delta, which utilizes only the control group's standard deviation, is an additional indicator when each group has a different standard deviation. Hedges' g, which gives a weighted estimate of the effect size by the relative size of each study, is an alternative having dissimilar sample sizes.

5.3. Wilcoxon Test

The following null and alternative hypotheses need to be tested:

H0. *Median (Difference)* $\leq$ *0.*

H1. *Median (Difference) > 0.*

Observe that the sample size n = 41 > 30 is large enough to use normal approximation, so a z-statistic will be used. The significance level, based on the information provided is alpha = 0.05, and the critical value for a right-tailed test is Zc = 1.64. The rejection region for this right-tailed test is R = {z:z > 1.64} R = z:z > 1.64. The z-statistic is computed as follows

$$z = \text{T} - \text{n}\ (\text{n} + 1)/4\text{nn} + 12\text{n} + 1/(24) \tag{5}$$

The given table shows the test results using Cohen's d, Glass's delta, and Hedge's g results.

The given table shows the test results using Wilcoxon signed rank test results.

5.4. Analysis of Validation Test

The results of validation test shown in Table 7 using Cohen's d, Glass's delta, and Hedges' g test for the research experiment to validate the experiment conducted for the experiment. For multi-class KNN filter (all, ckloc, IG) base paper, we have z-value = (−5.501, −5.579, −5.579), Cohen's d = (2.537, 3.180, 2.846), *p*-value = (1.00, 1.00, 1.00), which are greater than α = 0.05. For the ANN filter (all, ckloc, IG) base paper, we get *p*-value (1.00, 1.00, 0.627), Cohen's d = (2.232, 2.111, 0.064), and z-value (−5.579, −5.216, −0.324). For the validation test results of search-based optimizer RFE (all, ckloc, IG) we have *p*-value = (0.00261, 0.00564, 0.00013), Cohen's d = (3.432, 2.868, 0.586), and z value = (−5.566, −5.566, −2.287). The validation results of our classifier (all, ckloc, IG) base paper, we get *p*-value = (1.00, 1.00, 0.992), Cohen's d = (4.874, 5.302, 0.811), and z-value is (−5.579, −5.579, −3.155). It is concluded that it rejects the null H0 hypothesis. There is also insufficient evidence to say that the population mean of variations is greater than zero at the significance level of alpha = 0. 05α = 0.05. For validation test of our paper, we carry out the Wilcoxon signed rank test for filters, optimizer, and classifiers. We get the *p*-value for KNN vs. ANN filter (all, ckloc) to be (0.00364, 0.00217), *p*-value for RFE vs. GA (all, ckloc, IG) to be (0.00261, 0.00564, 0.00013), and *p*-value for classifiers vs. NB (ckloc, IG) to be (0.00024, 0.00128). It is concluded that it rejects the null H0 hypothesis which validates our results (referred to Tables 7 and 8).

Table 7. Cohen's d, Glass's delta, and Hedges' g results.

Algorithm	**Cohen's d**	**Glass's Delta**	**Hedges' g**
KNN (all)	2.537	2.195	2.537
KNN (ckloc)	3.180	2.725	3.180
KNN (IG)	2.846	2.391	2.846
ANN filter (all)	2.232	7.037	2.232
ANN filter (ckloc)	2.111	3.605	2.111
ANN filter (IG)	0.064	0.056	0.064
RFE (all)	3.432	7.751	3.432
RFE (ckloc)	2.868	4.567	2.868
RFE (IG)	0.586	0.531	0.586
Classifier (all)	4.874	5.254	4.874
Classifier (ckloc)	5.302	5.148	5.302
Classifier (IG)	0.811	0.634	0.811

Table 8. Wilcoxon signed rank test results.

Algorithm	***p*-Value (Wilcoxon Test)**
KNN vs. ANN filter-all	0.00364
KNN vs. ANN filter-ckloc	0.00217
RFE vs. GA-all	0.00261
RFE vs. GA-ckloc	0.00564
RFE vs. GA-IG	0.00013
Classifier vs. NB-ckloc	0.00024
Classifier vs. NB-IG	0.00128

6. Threats to Validity

Throughout an empirical study, one should be responsive of the possible threats to the legitimacy of the obtained results and findings that are derived from the experiment. The possible threats to the validity recognized for this research are evaluated in two categories, namely: construct, external and conclusion validity.

6.1. Construct Validity

In this experiment, SCM, OO, and LOC are the only metrics used from the datasets. These measures were commonly used in previous studies. Although these metrics can achieve good performance, but the usefulness of this metrics has been widely criticized. There are still some errors that might exist in non-defective labels as not all the defects were detected. This could be a potential threat to preparation and assessment of defect prediction models. Moreover, we have not checked for different values of iterations in ANN filter. We only chose specific elements for large datasets. Based on 1000 iterations, one might notice performance changes in terms of prediction.

6.2. External Validity

Our experiment is restricted to the evidence and background studies, therefore, specific findings from research are difficult to draw. All the projects that lead to our research are written in JAVA and that would certainly affect the generalization of our results. Although many studies have used subsets of our used databases as the foundation for their results, there is no guarantee that conclusions drawn from those ventures will be generalized. Mainly the applicability of the findings to industrial and closed source applications could be specific as there are typically more strict levels of code quality involved with such initiatives.

6.3. Conclusion Validity

Wilcoxon signed rank test need more post hoc examination to classify the localities of variations observed in multiple groups. We carry out Wilcoxon experiments in pairs to find possible differences of comparisons for different versions and other CPDPs. Additionally, Cohen's d for similar studies was used as effect size to measure the extent of the discrepancies. Another challenge is the lack of test of assessment. Certain studies may consider different steps to test the methodology and some of the findings and results might alter as a result. While our method works well for a large portion of the datasets (as opposed to CPDP base papers), it is not necessarily better for all of them, and further investigation is needed.

7. Conclusions

As per our research analysis, the optimizer and classifier are dependent upon the nature of the classes of data i.e., multi or binary. The base paper treated the cross-project defect repository of Tera PROMISE as binary class data whereas through EDA, it is revealed that data are multi-class in nature. There was variation in the features selected in our experiment through info-gain and selected in base paper. We then applied the same filter using multi-class data and obtained significant results. However, ANN filter still has more significant results as compared with the KNN filter. We then used RFE as a search-based optimizer and get better results than the given base paper. We used a classifier for our multi class dataset and our results improved by 30% as compared with base paper. The results are significantly improved by 40% to 60% using filters and we analyzed those results are also improved using multi-class. Our proposed method outperforms for the base paper of CPDP using multi-class. It seems that using ckloc features the results are high as compared with IG and all features. There is huge variation in results using IG features therefore, it is not recommended for future studies. We can use ckloc features for further investigation to improve the quality and performance of data collection in preparation. The success of our selection method will be affected by the amount of training sets for candidates. We intend to examine further possible determinants for effective cross-project detection of defects and to establish more reliable and accurate methods for selecting data for testing. In future, we can predict as categorical class for the base paper of CPDP, which is ultimately advantageous to predict defects on early basis. The cost and resources will be less using this approach and training sets will be available for further analysis.

Author Contributions: Conceptualization, R.b.F., S.S., M.S. and H.T.R.; Methodology, R.b.F. and S.S.; Software, R.b.F. and S.S.; Validation, H.T.R.; Formal analysis, R.b.F.; Investigation, M.S.; Resources, H.T.R.; Data curation, S.S.; Writing—original draft, R.b.F., S.S. and M.S.; Writing—review & editing, R.b.F., S.S., M.S. and H.T.R.; Visualization, R.b.F., S.S., M.S. and H.T.R.; Supervision, H.T.R.; Funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to King Saud University, Saudi Arabia, for funding this work through Researchers Supporting Project number (RSPD2023R704), King Saud University, Riyadh, Saudi Arabia.

Data Availability Statement: Data sharing is not applicable—no new data are generated.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Herbold, S.; Trautsch, A.; Grabowski, J. Global vs. local models for cross project defect prediction: A replication study. *Empir. Softw. Eng.* **2017**, *22*, 1866–1902. [CrossRef]
2. Zimmermann, T.; Nagappan, N.; Gall, H.; Giger, E.; Murphy, B. Cross project defect prediction: A large scale experiment on data vs. domain vs. process. In Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Amsterdam, The Netherlands, 24–28 August 2009; pp. 91–100. [CrossRef]
3. Basili, R.V.; Briand, L.; Melo, L.W. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Softw. Eng.* **1996**, *22*, 751–761. [CrossRef]

4. Yu, Q.; Qian, J.; Jiang, S.; Wu, Z.; Zhang, G. An Empirical Study on the Effectiveness of Feature Selection for Cross Project Defect Prediction. *IEEE Access* **2019**, *7*, 35710–35718. [CrossRef]
5. Moser, R.; Pedrycz, W.; Succi, G. A Comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, 10–18 May 2008; pp. 181–190. [CrossRef]
6. Ostrand, T.J.; Weyuker, E.J.; Bell, R.M. Predicting the location and number of faults in large software systems. *IEEE Trans. Softw. Eng.* **2005**, *31*, 340–355. [CrossRef]
7. Hosseini, S.; Turhan, B.; Mäntylä, M. A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction. *Inf. Softw. Technol.* **2018**, *95*, 296–312. [CrossRef]
8. Ryu, D.; Jang, J.I.; Baik, J. A transfer cost-sensitive boosting approach for cross-project defect prediction. *Softw. Qual. J.* **2017**, *25*, 235–272. [CrossRef]
9. Shukla, S.; Radhakrishnan, T.; Muthukumaran, K.; Neti, L.B.M. Multi-objective cross-version defect prediction. *Soft Comput.* **2018**, *22*, 1959–1980. [CrossRef]
10. Zhang, F.; Zheng, Q.; Zou, Y.; Hassan, A.E. Cross-project defect prediction using a connectivity-based unsupervised classifier. In Proceedings of the IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, USA, 14–22 May 2016; pp. 309–320. [CrossRef]
11. Turabieh, H.; Mafarja, M.; Li, X. Iterated feature selection algorithms with layered recurrent neural network for software fault prediction. *Expert Syst. Appl.* **2019**, *122*, 27–42. [CrossRef]
12. Cheikhi, L.; Abran, A. Promise and ISBSG Software Engineering Data Repositories: A Survey. In Proceedings of the 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, Ankara, Turkey, 23–26 October 2013; pp. 17–24. [CrossRef]
13. Zhang, F.; Keivanloo, I.; Zou, Y. Data Transformation in Cross-project Defect Prediction. *Empir. Softw. Eng.* **2018**, *22*, 3186–3218. [CrossRef]
14. Wu, F. Empirical validation of object-oriented metrics on NASA for fault prediction. *Commun. Comput. Inf. Sci.* **2011**, *201*, 168–175. [CrossRef]
15. Turhan, B.; Menzies, T.; Bener, A.B.; Stefano, J.D. On the relative value of cross-company and within-company data for defect prediction. *Empir. Softw. Eng.* **2009**, *14*, 540–578. [CrossRef]
16. Gong, L.; Jiang, S.; Bo, L.; Jiang, L.; Qian, J. A Novel Class-Imbalance Learning Approach for Both Within-Project and Cross Project Defect Prediction. *IEEE Trans. Reliab.* **2020**, *69*, 40–54. [CrossRef]
17. Li, Y.; Huang, Z.; Wang, Y.; Fang, B. Evaluating data filter on cross-project defect prediction: Comparison and improvements. *IEEE Access* **2017**, *5*, 25646–25656. [CrossRef]
18. Laradji, I.H.; Alshayeb, M.; Ghouti, L. Software defect prediction using ensemble learning on selected features. *Inf. Softw. Technol.* **2015**, *58*, 388–402. [CrossRef]
19. Hammouri, A.; Hammad, M.; Alnabhan, M.; Alsarayrah, F. Software Bug Prediction using machine learning approach. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 78–83. [CrossRef]
20. Chen, X.; Shen, Y.; Cui, Z.; Ju, X. Applying Feature Selection to Software Defect Prediction Using Multi-Objective Optimization. In Proceedings of the IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 4–8 July 2017; Volume 2, pp. 54–59. [CrossRef]
21. Mohammadi, S.; Mirvaziri, H.; Ghazizadeh-Ahsaee, M.; Karimipourb, H. Cyber intrusion detection by combined feature selection algorithm. *J. Inf. Secur. Appl.* **2019**, *44*, 80–88. [CrossRef]
22. Goel, L.; Sharma, M.; Khatri, S.; Damodaran, D. Prediction of Cross Project Defects using Ensemble based Multinomial Classifier. *EAI Endorsed Trans. Scalable Inf. Syst.* **2019**, *7*, e5. [CrossRef]
23. Bommert, A.; Sun, X.; Bischl, B.; Rahnenführer, J.; Lang, M. Benchmark for filter methods for feature selection in high-dimensional classification data. *Comput. Stat. Data Anal.* **2020**, *143*, 106839. [CrossRef]
24. Jalil, A.; Faiz, R.B.; Alyahya, S.; Maddeh, M. Impact of Optimal Feature Selection Using Hybrid Method for a Multiclass Problem in Cross Project Defect Prediction. *Appl. Sci.* **2022**, *12*, 12167. [CrossRef]
25. He, Z.; Shu, F.; Yang, Y.; Li, M.; Wang, Q. An investigation on the feasibility of cross-project defect prediction. *Autom. Softw. Eng.* **2012**, *19*, 167–199. [CrossRef]
26. Giray, G.; Bennin, K.E.; Köksal, Ö.; Babur, Ö.; Tekinerdogan, B. On the use of deep learning in software defect prediction. *J. Syst. Softw.* **2023**, *195*, 111537. [CrossRef]