applied sciences

MDPI

Article

Performance Evaluation of Convolutional Neural Network for Multi-Class in Cross Project Defect Prediction

Sundas Noreen [1,*], Rizwan Bin Faiz [1], Sultan Alyahya [2] and Mohamed Maddeh [3]

[1] Faculty of Computing, Riphah International University, Islamabad 46000, Pakistan
[2] Department of Information Systems, King Saud University, Ar Riyadh 12372, Saudi Arabia
[3] College of Applied Computer Science, King Saud University, Ar Riyadh 12372, Saudi Arabia
* Correspondence: sundasnoreen12@gmail.com

Abstract: Cross-project defect prediction (CPDP) is a practical approach for finding software defects in projects which have incomplete or fewer data. Improvements to the defect prediction accuracy of CPDP—such as the PROMISE repository, the correct classification of the source data, removing the noise, reducing the distribution gap, and balancing the output classes—are an ongoing challenge, as is the selection of an optimal feature set. This research paper aims to achieve a higher defect prediction accuracy for multi-class CPDP by selecting an optimal feature set through XGBoost combined with an automatic feature extraction using a convolutional neural network (CNN). This research type is explanatory, and this research method is controlled experimentation, for which the independent variable prediction accuracy was dependent upon two variables, XGBoost and CNN. The Softmax layer was added to the output layers of the CNN classifier to classify the output into multiple classes. In our experimentation with CPDP, we selected all 28 versions of the multi-class, in which 11 versions were selected as the source projects, against which we predicted 28 target versions with an average AUC of 75.57%. We validated this research paper's results through the Wilcoxon test. Therefore, after removing the noise, class imbalances, and the data distribution gap, and treating the PROMISE dataset as multi-class, the optimal features selected through XGBoost and classified through the CNN can substantially increase the prediction accuracy in CPDP as evident from our exploratory data analysis (EDA).

Keywords: cross-project defect prediction; extreme gradient boosting (XGBoost); convolutional neural network (CNN); CTGAN

check for updates

Citation: Noreen, S.; Faiz, R.B.; Alyahya, S.; Maddeh, M. Performance Evaluation of Convolutional Neural Network for Multi-Class in Cross Project Defect Prediction. *Appl. Sci.* **2022**, *12*, 12269. https://doi.org/10.3390/app122312269

Academic Editors: Neal N. Xiong and Muhammad Imran

Received: 9 October 2022
Accepted: 14 November 2022
Published: 30 November 2022

1. Introduction

Nowadays, software project success is a major challenge. A major headache for project managers is defects or bugs. These bugs arise due to a bad design and the implementation of code [1]. Software defect prediction (SDP) can precisely find defects in the initial stages of software development. It emphasizes recognizing defect tendencies in software modules and helps researchers allocate limited resources to modules with high possibilities of comprising defects [2–4]. SDP can solve the problems of inadequate developer energy and limited development cycles; on the other hand, it can effectually improve the quality of the software [5,6].

The significance of software reliability has risen due to the increasing complexity of software [4]. A substantial amount of testing and debugging is compulsory for constructing reliable software. As budget and time are two limitations, efforts need to be ranked for productivity. To overcome these above-mentioned issues, software defect prediction techniques [7] have been extensively used to help developers in ranking their efforts for testing and debugging by predicting the occurrence of bugs [6].

SDP can be divided into two broad categories: within-project defect prediction (WPDP) [4,8–12] and cross-project defect prediction (CPDP) [12–15]. The limitation for

WPDP when building a realistic defect prediction model is a lack of code information in the early stages whereas, for CPDP, sufficient code information from mature projects can be used to train the model, which can predict whether the new file of the target project has defects or not [11].

However, in practice, achieving satisfactory CPDP is challenging. Both the source and the target projects have a significant distribution gap [12–15], which violates the basic requirements of most machine learning modeling technology, such as the similar distribution assumption. To address this issue, we removed all the instances between the source and target projects which violate the distribution gap requirement using min–max normalization, hence bringing the source and target projects to the same pace.

There is a considerable class imbalance issue that exists between two different projects which adversely affects the accuracy of CPDP. To learn good feature representation and resolve this issue, we used the conditional tabular generative adversarial network (CTGAN). The CTGAN is a collection of deep-learning-based synthetic data generators for tabular data, which can learn from real data and generate synthetic clones with a high fidelity.

Feature selection is the most powerful technique in machine learning. It can eliminate irrelevant and redundant features in the defect datasets and have a diverse impact on the model's prediction accuracy [3,5]. We used XGBoost for the feature selection. XGBoost is an influential machine learning algorithm. XGBoost continuously reduces the loss of the previously generated decision tree. It generates a new decision tree by continuously updating the weights of the provided sample data. The weights of the different metrics are then averaged out to determine the importance of each metric from the provided sample data [1]. XGBoost performed very well on our model, so much so that we achieved an accuracy of 88%.

In recent years, many studies [16–19] have proposed extracting hidden semantic meaning through deep learning (DL). Feature extraction methods reduce the number of features by creating new, combined features from the original features [3,5]. We have clear evidence that the pooling layer of CNNs have an automatic feature extraction capability which outperforms the classification results. So, we used the CNN for an automatic feature extraction along with the classifier.

The limitation for WPDP when building a realistic defect prediction model in the early stages occurs because of a lack of code information for a better resource utilization in terms of the team, budget, and time. To overcome this problem, researchers came up with the idea of cross-project defect prediction. By working across projects, enough code information from a mature project can be used to train the model, which can then predict whether the new file of the target project is defective or not. We intend to observe the performance of the CNN model, given its automatic feature extraction and multi-class classification abilities, by evaluating the defect prediction accuracy using the PROMISE repository. The CNN pooling layer plays an important role in an automatic quadratic feature extraction from both the source and target projects.

It was evident from the EDA that the PROMISE repository [20] is multi-class and has distribution gaps between the different projects' datasets, noise, and class imbalances. Our experimental results show that, after removing the noise, reducing the distribution gaps, and balancing the output classes, the optimal feature set is selected through XGBoost, and a better feature extraction is achieved through the pooling layer of the CNN. Our experimental results show a high prediction accuracy cross-project for multi-class data through a CNN classifier when combined with the softmax.

2. Literature Review

Cross-project defect prediction (CPDP) has recently drawn great interest. To predict defects in projects without sufficient training data for CPDP, many researchers tried to build innovative and competitive CPDP models. However, not all studies report good performances of CPDP. Many of the studies reviewed in the literature have established software defect prediction models for estimating the prediction regarding the software development process including effort estimation, resource utilization, and maintainability during the software development lifecycle.

(Y. SUN, X.-Y. JING, F. WU, J. J. LI, D. XING, H. CHEN and Y. SUN) [11] worked on a cross-project defect prediction using the PROMISE repository. They normalized the data by using the commonly used min–max normalization. They have done feature selection by randomly selecting the features with the same metrics from the datasets. They used a four layered neural network classifier to predict the class of the mapped instances and added Softmax activation on the top of the neural network for a multi-class classification. They constructed the experiment and the result analyses have demonstrated the effectiveness and efficiency of the proposed approach with the G-measure to be 0.66. However, they did not normalize the dataset in terms of the noise [7,21]. They used all the features to train the classifier instead of selecting the most optimal features set [7] that has more impact on the prediction accuracy.

In a series of experiments, (LEI SHENG, LU LU & JUNHAO LIN, 2020) [22] proposed the ADCNN model to extract the important features by the CNN. The ADCNN can learn the semantic and structural features from the source code directly and reduce the distribution gap between the source and target projects in order to achieve a higher defect prediction in the cross-project. They solved the issue of noise for a better data normalization. They divide the output into two classes by using the classification method such as logistic regression. They measured the overall experimental results in terms of the F-measure, AUC, and PofB20. However, they treated the data set into a binary class instead of a multi-class, whereas it is clearly evident from the existing literature [11] that the PROMISE dataset is multi-class. They also used all the features to train the classifier instead of selecting the most optimal features set [6] that has more impact on the prediction accuracy.

(SEYEDREBVAR HOSSEINI A, BURAK TURHAN B & MIKA MÄNTYLÄA, 2017) [21] proposed a model and showed the effectiveness of a feature selection by info gain on the defect prediction accuracy in the cross-project. They aimed to converge to an optimal training dataset and at the same time, they considered the effect of a feature selection and the potential noise in the labeling of the datasets. They classify the output into two classes by using the Naïve Bayes (NB) classifier. However, they did not normalize the dataset in terms of the class imbalance [7,11,21] and distribution gap [7,11,21]. They also treated the data set into a binary class instead of a multi-class, whereas it is clearly evident from the existing literature [11] that the PROMISE dataset is multi-class.

(CONG PAN, MINYAN LU, BIAO XU AND HOULENG GAO, 2020) [7] proposed an improved CNN model which could be able to better learn semantic representations from source-code ASTs for WPDP. They resolved the issue of noise for a data normalization and performed a feature selection. They divided the output into binary classes by using the logistic regression (LR) classifier. The statistical hypothesis test showed that their method was almost significantly better than other machine learning models in terms of the evaluation metrics, such as the F-Measure and G-Measure. However, they did not normalize the dataset in terms of the class imbalance [7,11,21], noise [7,21], and the distribution gap [7,11,21]. They also treated the data set into a binary class instead of a multi-class, whereas it is clearly evident from the existing literature [11] that the PROMISE dataset is multi-class. They used all the features to train the classifier instead of selecting the most optimal features set [7] that has more impact on the prediction accuracy.

(JIE ZHANG (Graduate Student Member, IEEE), JIA JING WU (Senior Member, IEEE), CHUAN CHEN (Member, IEEE), ZIBIN ZHENG (Senior Member, IEEE), and MICHAEL R. LYU (Fellow, IEEE), 2020) [23] proposed a cross version software defect prediction model with data selection. They proposed a novel clustering-based multi-version classifier (CMVC), which can automatically select the most relevant training data by assigning higher weights to those versions which have less noise than others. They classified the output of the classifier into two classes such as clean or buggy. They used the F-Measure as an evaluation metric. However, they did not normalize the dataset in terms of the class imbalance [7,11,21], noise [7,21], and the distribution gap [7,11,21]. They also treated the data set into a binary class instead of a multi-class, whereas it is clearly evident from the existing literature [11] that the PROMISE dataset is multi-class. They used all the features to train the classifier instead of selecting the most optimal features set [7] that has more impact on the prediction accuracy.

(AILI WANG, YUTONG ZHANG, HAIBIN WU, KAIYUAN JIANG, AND MINHUI WANG, 2020) [1] proposed few-shot learning-based balanced distribution adaptation for heterogeneous defect prediction. They removed the distribution gap between the source and target projects of the NASA repository. They used XGBoost to select the important features before training the classifier. They classified the output of the classifier into binary classes such as clean or buggy. They used AdaBoost as a classifier. They measured their results in terms of the AUC, G, and F measure. However, they did not normalize the dataset in terms of noise [7,21]. They also treated the data set into a binary class instead of a multi-class, whereas it is clearly evident from the existing literature [11] that the PROMISE dataset is multi-class.

Many studies have focused on the issue of attaining a higher accuracy in terms of a cross-project defect prediction. While much research has been conducted on achieving a higher cross-project defect prediction accuracy, only a few researchers have taken the class imbalance, noise, feature selection, and feature extraction together into consideration. Table 1 describes the research summary of our literature review.

Table 1. Research summary.

Sr #	**Reposit Ory**	**Data Pre-Processing**			**Feature Selection**	**Classes**		**Classification Method**	**Statistical Test**	**Measure**
		Class Imbalance	**Noise**	**Data Distribution**		**Multi Class**	**Binary Class**			
1	PROMISE	✓	✗	✓	All	✓	✗	NN	✗	0.66
2	PROMISE	✓	✓	✓	All	✗	✓	CNN	Scott Knott ESD test	0.66
3	PROMISE	✗	✓	✗	Info gain	✗	✓	Naïve Byes	Wilcoxon and Kruskal–Wallis H test	0.47
4	PROMISE	✗	✗	✗	All	✗	✓	CNN	Friedman test	0.61
5	PROMISE	✗	✗	✗	All	✗	✓	Random forest, logistic regression and Naïve Bayes	Wilcoxon signed rank test and cliff's delta	0.61
6	NASA	✓	✗	✓	XGBoost	✗	✓	Ada Boost	—	0.94

We built this research paper gap based on our literature review and its summary, which is summarized in Table 1.

The research gaps of our detailed study are as follows:

Since it is obvious after the EDA that the PROMISE repository is multiclass [11] and by reducing the distribution gap between the projects [7,11,21], removing the noise [7,21], and resolving the class imbalance [7,11,21], a higher cross-project defect prediction accuracy can be achieved.

It is also obvious from the existing literature that a higher defect prediction accuracy can be achieved in a cross-project by selecting the optimal features set through XGBoost [7], since XGBoost helps to select the optimal features set.

It is obvious from the existing literature [21] that the CNNs property of an auto feature extraction by its pooling layers can help achieve a higher defect prediction accuracy. In addition, it is also obvious from the existing literature that a Softmax layer is better suited for multi classification [11] and has not yet been experimented with the CNN as a classifier in CPDP.

3. Research Methodology

This research paper type is explanatory. Explanatory research aims to explain the causes and consequences of a well-defined problem. If the cross-project defect prediction is a well-defined problem, we will find the defect prediction accuracy using deep learning techniques such as the CNN. This research paper will use experimental research methods to manipulate and control the variables to determine the cause and effect between the variables for the prediction accuracy and authenticity. The perspective of our experiment is an earlier defect prediction based on already developed mature projects. We will predict the defects earlier by training the classifier using mature source projects and then predict the defects in the target projects.

The purpose of our experiment is a performance evaluation of the convolution neural networks in cross-project defect prediction. Improving the defect prediction accuracy can be done using a convolution neural network in cross-project defect prediction by removing the noise, resolving the class imbalance, reducing the distribution gap, and selecting the important features. The objects of our study are the source and target projects of the PROMISE repository. This research paper will use the Wilcoxon test for ascertaining the cross-project defect prediction's authenticity as a statistical analysis. According to our literature review, research summary, research gap, and research objective, we will be addressing the following questions in this research paper:

RQ1: What is the impact of XGBoost for optimal feature selection for multi-class in predicting the cross-project defect prediction accuracy of PROMISE repository?

H0. *XGBoost does not select optimal feature sets for multi-class in predicting the cross-project defect prediction accuracy of PROMISE repository.*

H1. *XGBoost selects the optimal feature sets for multi-class in predicting the cross-project defect prediction accuracy of PROMISE repository.*

- **Dependent & Independent variables**

This research paper has the CNN as the change/manipulate independent variables and the accuracy (AUC) as the dependent variables.

- **Design**

This research paper design is IF1T (1 factor 1 treatments).

○ Factor-design method.
○ Treatments.

1. CNN

RQ2: What is the impact of the CNN as a classifier in combination with the Softmax for the multi-class in predicting the cross-project defect prediction accuracy of PROMISE repository?

H0. *The CNN classifier in combination with Softmax has no impact for the multi-class in predicting the cross-project defect prediction accuracy of the PROMISE repository.*

H1. *The CNN as a classifier in combination with Softmax has an impact for the multi-class in predicting the cross-project defect prediction accuracy of PROMISE repository.*

- **Dependent & Independent variables**

This research paper has XGBoost as the change/manipulate independent variables and the accuracy (AUC) as the dependent variables.

- **Design**

This research paper design is IF1T (1 factor 1 treatments).

- **Factor**-design method.
- **Treatments**-XGBoost.

4. Materials and Methods

In this section, we propose our CPDP model. Figure 1 visualizes the whole research process reported in this paper. We will train our model using the mature projects of the PROMISE repository and will predict the defects prediction accuracy on the target projects. By following this approach, we will detect the defects earlier in the target projects and will allocate time and resources to the faulty areas accordingly.

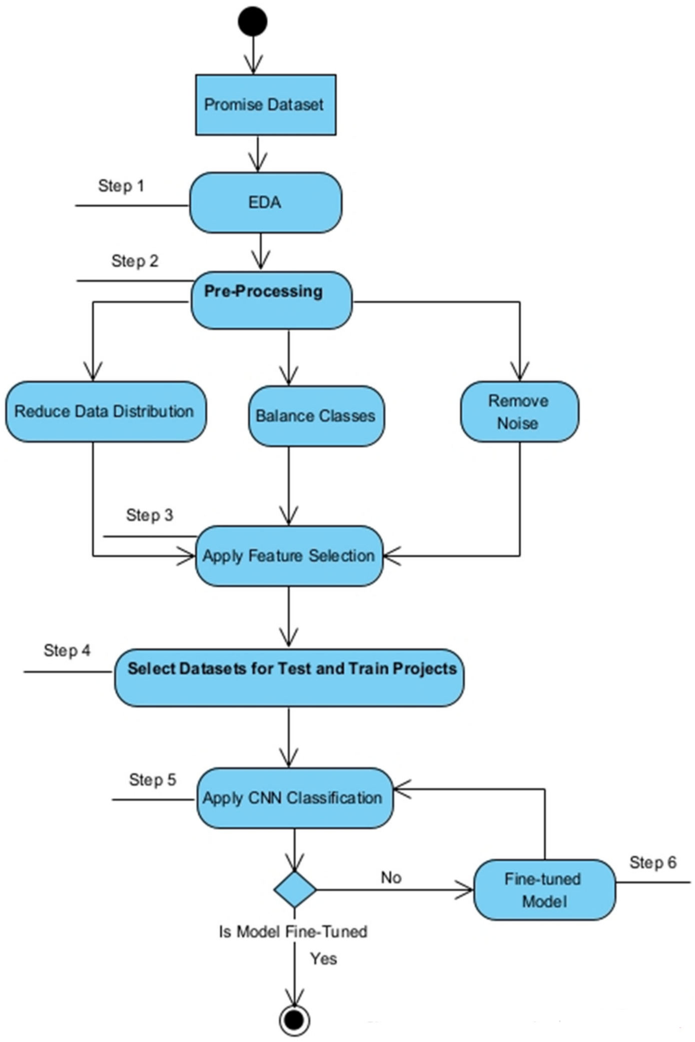


Figure 1. Our proposed methodology.

4.1. Exploratory Data Analysis (Step 1)

We performed exploratory data analysis (EDA) on the PROMISE repository from the year 2020. The essential purpose of EDA is spotting the missing and erroneous data, mapping and understanding the underlying structure of the data, and identifying the most important variables in the dataset.

The PROMISE is an open-source repository that is widely used for defect prediction studies. The PROMISE repository has a total of 45 versions consisting of 11 different projects. Among these 45 projects, we have ignored all those versions of projects whose output is binary class, as we are conducting an experiment on a multi-class problem. After removing the binary projects, we are left with 35 versions of 11 projects. These projects include Ant, Camel, Ivy, Jedit, Log4j, Lucene, Poi, Synapse, Velocity, Xalan, and

Xerces. Each version of the project contains 20 features that are described in Table 2. These object-oriented attributes are used as the independent variables and bugs are used as the dependent variable representing the defective class.

Table 2. Promise repository features along with their description.

Attribute Full Form [7]	Attribute [20]	Description
Weighted methods per class	WMC	The number of methods used in each class [24]
Depth of inheritance tree	DIT	The maximum distance from a given class to the root of an inheritance tree [24]
Number of children	NOC	The number of children of a given class in an inheritance tree [24]
Coupling between object classes	CBO	The number of classes that are coupled to a given class [24]
Response for a class	RFC	The number of distinct methods invoked by code in each class [24]
Lack of cohesion in methods	LCOM	The number of method pairs in a class that do not share access to any class attributes [24]
Afferent couplings	CA	Afferent coupling, which measures the number of classes that depends upon a given class [25]
Efferent couplings	CE	Efferent coupling, which measures the number of classes that a given class depends upon [25]
Number of public methods	NPM	The number of public methods in each class [25]
Lack of cohesion in methods	LCOM3	Another type of lcom metric proposed by Henderson-Sellers [26]
Lines of code	LOC	The number of lines of code in each class [26]
Data access metric	DAM	The ratio of the number of private/protected attributes to the total number of attributes in each class [26]
Measure of aggregation	MOA	The number of attributes in each class which are of user-defined types [26]
Measure of functional abstraction	MFA	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 [26]
Cohesion among methods of class	CAM	The ratio of the sum of the number of different parameter types of every method in each 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 [26]
Inheritance coupling	IC	The number of parent classes that a given class is coupled to [26]
Coupling between methods	CBM	The total number of new or overwritten methods that all inherited methods in each class are coupled to [26]
Average method complexity	AMC	The average size of methods in each class [26]
Greatest value of CC	MAX_CC	The maximum McCabe's cyclomatic complexity (CC) score of methods in each class [26]
Arithmetic mean value of CC	AVG_CC	The arithmetic means of the McCabe's cyclomatic complexity (CC) scores of methods in each class [26]

After the EDA, we came to know that the PROMISE repository is of a multi-class instead of a binary class and has an issue of noise, a distribution gap, and a class imbalance. The following figures show the variation in the data along with the presence of multi-class data in multiple versions of the projects.

In Figures 2–4, the x-axis represents the class of the bug, and the y-axis represents the total number of rows for the respective class type. The bar height across each class mentioned the total number of instances for the specific class in Ant version 1.6, Xerces version 1.3, and Xerces version 1.4, respectively. From Figures 2–4, it is obvious that the PROMISE dataset is multi-class in nature, so, we treat the dataset as multi-class during the data normalization.

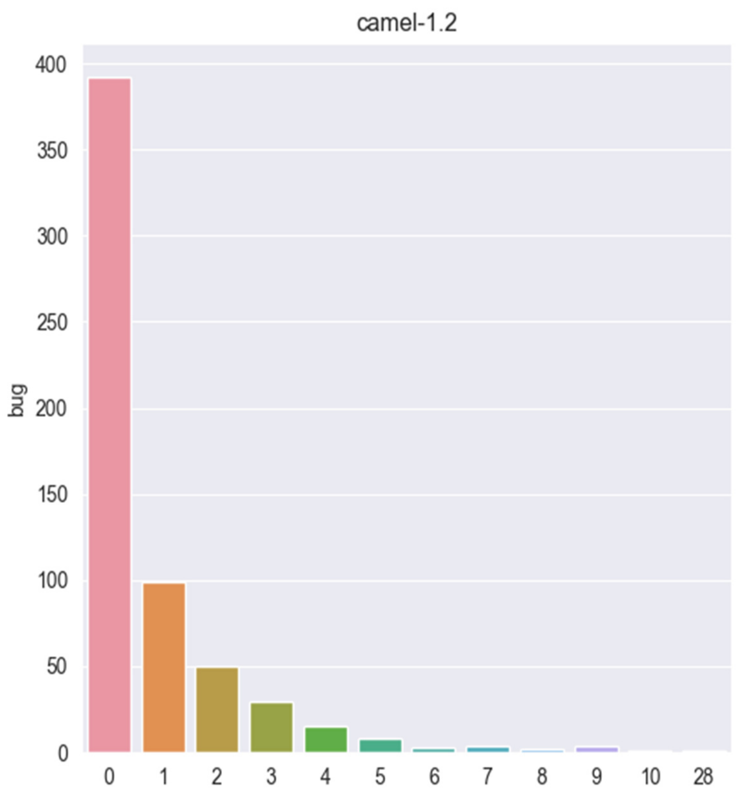


Figure 2. Multi-class of project Camel-1.2.

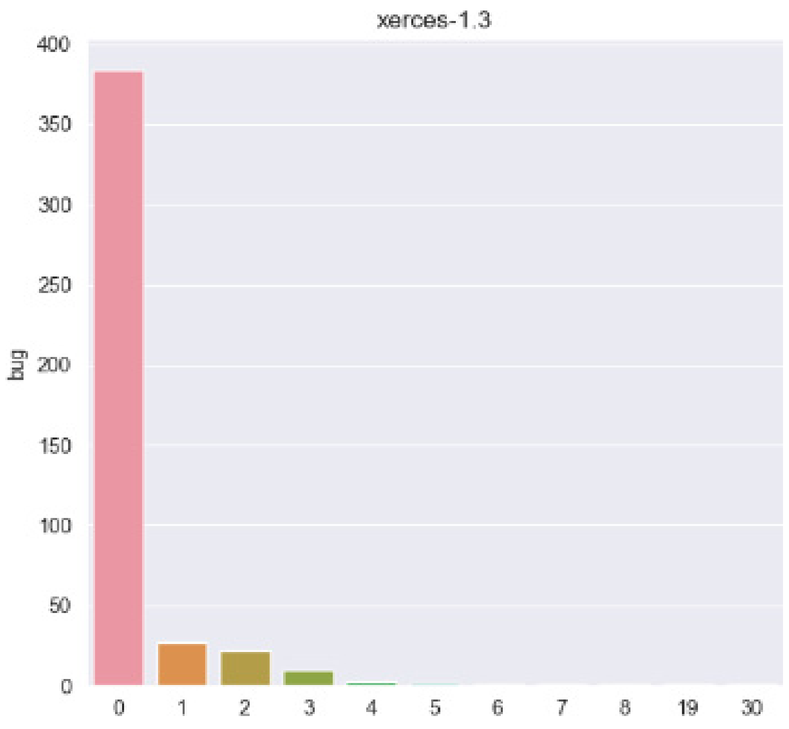


Figure 3. Multi-class of project Xerces 1.3.

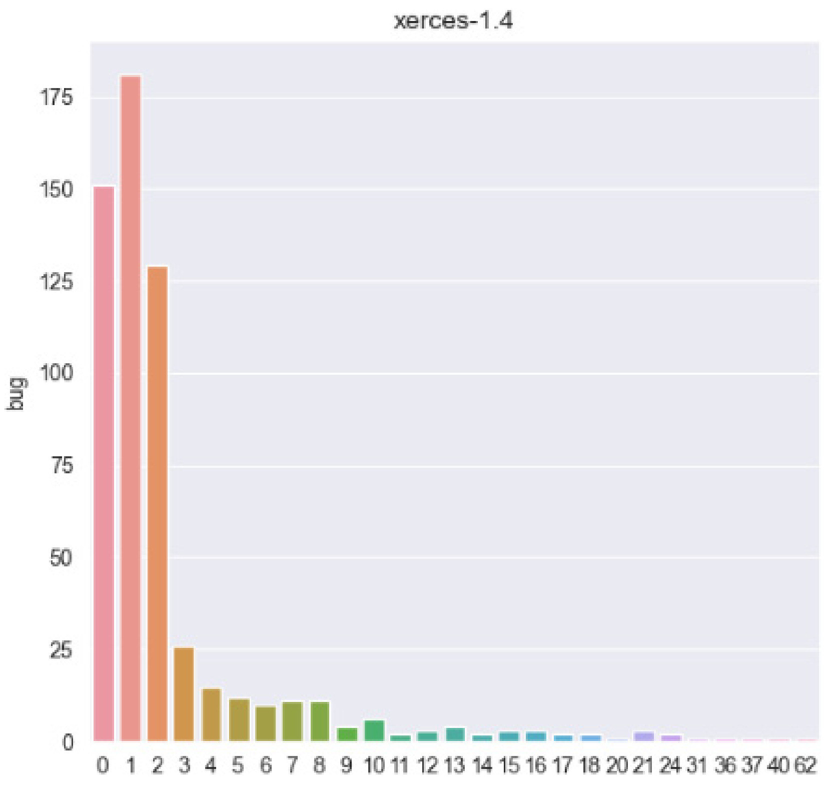
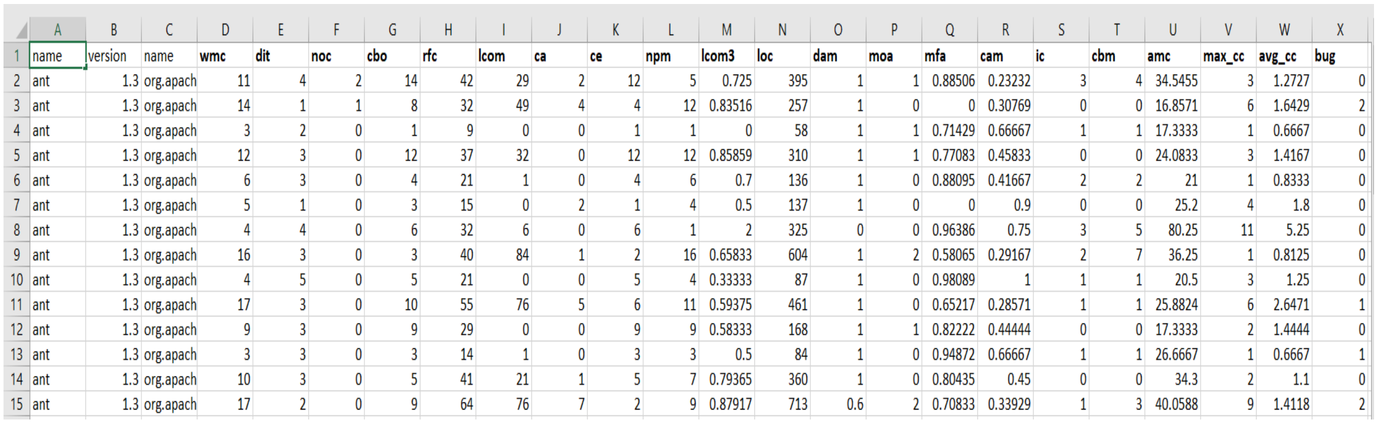


Figure 4. Multi-Class of project Xerces 1.4.

4.2. Data Pre-Processing (Step 2)

In Figure 5, the original dataset is displayed. The above figure describes the dataset of project Ant-1.3. The vertical columns describe the project features, and it consists of 20 features for each of the projects, and the horizontal rows describe the instance of the project.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	name	version	name	**wmc**	**dit**	**noc**	**cbo**	**rfc**	**lcom**	**ca**	**ce**	**npm**	**lcom3**	**loc**	**dam**	**moa**	**mfa**	**cam**	**ic**	**cbm**	**amc**	**max_cc**	**avg_cc**	**bug**
2	ant	1.3	org.apach	11	4	2	14	42	29	2	12	5	0.725	395	1	1	0.88506	0.23232	3	4	34.5455	3	1.2727	0
3	ant	1.3	org.apach	14	1	1	8	32	49	4	4	12	0.83516	257	1	0	0	0.30769	0	0	16.8571	6	1.6429	2
4	ant	1.3	org.apach	3	2	0	1	9	0	0	1	1	0	58	1	1	0.71429	0.66667	1	1	17.3333	1	0.6667	0
5	ant	1.3	org.apach	12	3	0	12	37	32	0	12	12	0.85859	310	1	1	0.77083	0.45833	0	0	24.0833	3	1.4167	0
6	ant	1.3	org.apach	6	3	0	4	21	1	0	4	6	0.7	136	1	0	0.88095	0.41667	2	2	21	1	0.8333	0
7	ant	1.3	org.apach	5	1	0	3	15	0	2	1	4	0.5	137	1	0	0	0.9	0	0	25.2	4	1.8	0
8	ant	1.3	org.apach	4	4	0	6	32	6	0	6	1	2	325	0	0	0.96386	0.75	3	5	80.25	11	5.25	0
9	ant	1.3	org.apach	16	3	0	3	40	84	1	2	16	0.65833	604	1	2	0.58065	0.29167	2	7	36.25	1	0.8125	0
10	ant	1.3	org.apach	4	5	0	5	21	0	0	5	4	0.33333	87	1	0	0.98089	1	1	1	20.5	3	1.25	0
11	ant	1.3	org.apach	17	3	0	10	55	76	5	6	11	0.59375	461	1	0	0.65217	0.28571	1	1	25.8824	6	2.6471	1
12	ant	1.3	org.apach	9	3	0	9	29	0	0	9	9	0.58333	168	1	1	0.82222	0.44444	0	0	17.3333	2	1.4444	0
13	ant	1.3	org.apach	3	3	0	3	14	1	0	3	3	0.5	84	1	0	0.94872	0.66667	1	1	26.6667	1	0.6667	1
14	ant	1.3	org.apach	10	3	0	5	41	21	1	5	7	0.79365	360	1	0	0.80435	0.45	0	0	34.3	2	1.1	0
15	ant	1.3	org.apach	17	2	0	9	64	76	7	2	9	0.87917	713	0.6	2	0.70833	0.33929	1	3	40.0588	9	1.4118	2

Figure 5. Noise in terms of duplicated records in version ant 1.3.

To answer and to describe each issue, we have divided our process into two steps for each which are:

- The method to perform the experiment.
- Display the dataset before and after performing the experiment.

4.2.1. Noise

The PROMISE repository has noise in terms of duplicated rows. We removed the noise from the PROMISE dataset by removing the redundant data, so that our trained model predicts the defects accurately with a higher accuracy.

- **Method to Perform Experiment** To resolve the issue of noise from the dataset, we performed following steps:
    - Read the CSV File.
    - Dropped all the unimportant columns such as the name and version.
    - Used pandas duplicated method on the data retrieved from the CSV file.
    - Duplicated method of panda's library traversed each row of the dataset one by one throughout the file and picked the duplicated rows.
    - Used pandas' method drop_duplicates on the data retrieved from the csv file.
    - Used the drop_duplicates method to remove all the retrieved duplicated rows from the dataset.
- **Display Dataset Before and After Performing Experiment** Tables 3 and 4 displays the result before and after performing the experiment.
    - **Before Removing Duplicated Rows**
    - **After Removing Duplicated Rows**

Table 3. Noise in terms of duplicated records in version ant 1.3.

	wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom3	loc	dam	moa	mfa	cam	ic	cbm	amc	Max_cc	Avg_cc	bug
81	2	3	0	3	7	1	1	3	2	2.0	23	0.0	0	0.969	1.0	0	0	10.5	1	0.5	1
96	2	3	0	3	7	1	1	3	2	2.0	23	0.0	0	0.969	1.0	0	0	10.5	1	0.5	1

Table 4. Removed noise in terms of duplicated records in version ant 1.3.

	wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom3	loc	dam	moa	mfa	cam	ic	cbm	amc	Max_cc	Avg_cc	bug
81	2	3	0	3	7	1	1	3	2	2.0	23	0.0	0	0.969	1.0	0	0	10.5	1	0.5	1
96	4	1	0	4	4	6	3	1	4	2.0	4	0.0	0	0.000	0.5	0	0	0.0	1	1.0	0

4.2.2. Distribution Gap

The PROMISE repository has an issue of a distribution gap as well between the source and the target projects which diversely affects the prediction accuracy of the trained model.

- **Method to Perform Experiment** To resolve the distribution gap from each dataset, we used min–max normalization, and we have performed following steps:
    - Read the source CSV file.
    - We created a list of all the columns.
    - We applied for loop for each column.
    - We calculated the minimum and maximum value based on the column.
    - We generated a list containing the minimum and maximum range value for each column.
    - We removed those values which are outside the calculated min and max acceptable columns values and made sure both the source and target comes on the same space.
- **Display Dataset Before and After Performing Experiment**

  Figures 6 and 7 displays the result before and after performing the experiment

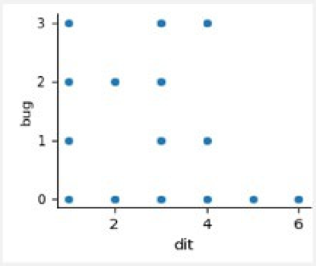
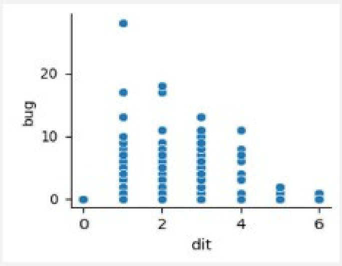
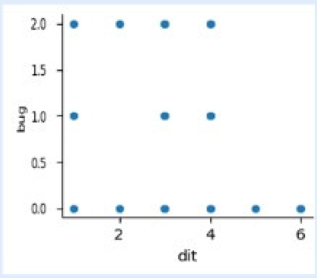
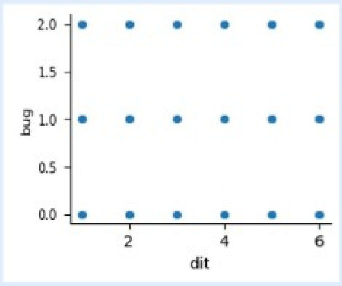
Features taken: dit	Source Project: Ant-1.3	Target Project: Camel
Before Covering Distribution Gap		
After Covering Distribution Gap		

Figure 6. Distribution gap before and after experiment by considering ant-1.3 as a source project and Camel as a target project in terms of the dit feature.

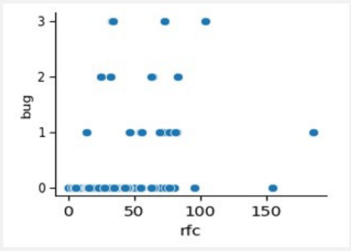
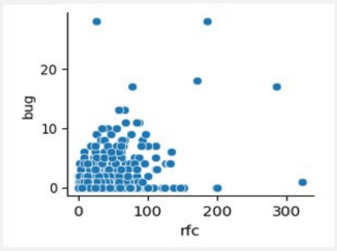
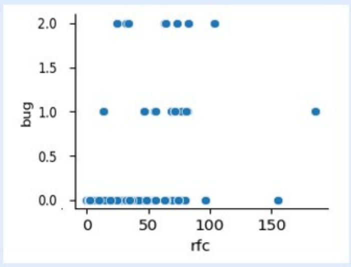
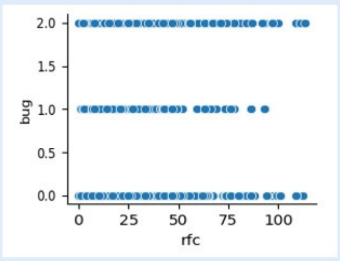
Features taken: rfc	Source Project: Ant-1.3	Target Project: Camel
Before Covering Distribution Gap		
After Covering Distribution Gap		

Figure 7. Distribution gap before and after experiment by considering ant-1.3 as a source project and Camel as a target project in terms of rfc feature.

In Figures 6 and 7, the horizontal scale mentions the ranges of the dit and rfc features for the Ant and Camel projects, and the vertical scale mentions the bug classes. We can see their distribution gap in terms of the ranges of the values for the dit and rfc features between the two projects before covering the distribution gap. We were unable to achieve a higher defect prediction accuracy by training the classifier using source projects and predicting defects in the target project. To achieve a higher defect prediction accuracy, we removed this distribution gap between the two projects using min–max normalization. After covering the distribution gap, the target project the dit and rfc features values lies under the range of the dit and rfc features values of the source project. By using this approach, we can achieve a higher defect prediction accuracy because both of the projects are now on the same space.

4.2.3. Class Imbalance

The PROMISE repository also has an issue of a class imbalance both in terms of the total number of instances and the total number of output classes. We solved the issue of a class imbalance using a CTGANSynthesizer.

- **Method to Perform Experiment** To solve the class imbalance issue from dataset, we performed the following steps:
  - Once the outliers had been removed from the dataset, we used a CTGANSynthesizer. The CTGAN is a collection of deep learning-based synthetic data generators for single table data, which can learn from real data and generate synthetic clones with a high fidelity.
  - We passed 100 as an epoch value as a parameter of the CTGANSynthesizer to get the most relevant values.
  - Then, we trained the CTGANSynthesizer model by passing the data and discrete columns as the parameters.
  - We passed the sample size as 960, the highest number of rows in the PROMISE repository datasets. Now using this technique, all the minority classes are oversampled to the majority class.
- **Display Dataset Before and After Performing Experiment**

  Tables 5 and 6 display the results before and after performing the experiment.

Table 5. List of total number of instances for each version before resolving class imbalance. Maximum number of rows are 960 for Camel-1.6 and minimum number of rows are 7 for forrest-0.6.

Class Name	Total Columns	Total Rows
Camel-1.6	24	960
Xalan-2.7	24	910
Xalan-2.6	24	886
Camel-1.4	24	873
Xalan-2.5	24	804
Ant-1.7	24	746
Forrest-0.8	24	33
Forrest-0.7	24	30
Pbeans1	24	27
Ckjm	24	11
Forrest-0.6	24	7

Table 6. List of total number of instances for each version after resolving class imbalance issue.

Class Name	Total Columns	Total Rows
Camel-1.6	24	960
Xalan-2.7	24	960
Xalan-2.6	24	960
Camel-1.4	24	960
Xalan-2.5	24	960
Ant-1.7	24	960
Forrest-0.8	24	960
Forrest-0.7	24	960
Pbeans1	24	960
Ckjm	24	960
Forrest-0.6	24	960

4.3. Feature Selection (Step 3)

The features contribute a lot in building a predictive model which has an excessive impact on the overall performance. Thus, it is important to intelligently select the optimal features that contribute to a high prediction accuracy, before training the model, to create a more comprehensive model having a low variance while training.

We used XGBoost to measure the significance of each feature using the characteristics of the classification and regression tree (CART). To lessen the cost of the segmented tree, the features having the highest weight are selected for segmentation until the maximum depth is reached. The gradient lifting algorithm is used continuously to lessen the loss of the previously generated decision tree, which diminishes the objective function and guarantees the trustworthiness of the final decision. The objective function helps to avoid an over-fitting [1].

We used XGBoost to select important features based on the importance given to each attribute. This technique helps to select the optimal features having a high dependency on the output.

- **Display Dataset Before and After Performing Experiment**

The PROMISE dataset has 20 features before the experiment and after applying XG-Boost, we were left with 15 features. XGBoost selected the LOC, NPM, RFC, CBO, WMC, LCOM, AMC, CA, CAM, MFA, AVG_CC, CE,

LCOM3, MAX_CC, and CBM as the optimal features based on their importance, as shown in Figure 8. LOC has a higher f score, which is measured as an average of its importance in all the subtrees.

4.4. Dataset Division (Step 4)

We selected the datasets as the source and target projects. We combined each version of the dataset as a compact dataset because of the less amount of data to train the classifier for a better prediction accuracy and used these combined versions as the source projects. We performed our experiment by selecting each version of every dataset as a target project to evaluate the prediction accuracy.

4.5. CNN Classification Model (Step 5)

Convolutional neural networks (CNN) are a specific kind of neural network for processing data. The convolutional network has been enormously effective in real-world applications. The name CNN specifies that the network works with a mathematical operation called convolution. The following are the positive aspects of the CNN:

- A major optimistic aspect of CNNs is auto feature extraction. The pooling layer plays an important part in extracting the important features from the input that contribute more to the final output. [6]
- CNNs are frequently exposed to pre-training, that is, to a process that initializes the network with pre-trained parameters instead of randomly set ones. Pre-training can quicken the learning process and boost the generalization ability of the network [6]
- The sparsity of the connections: in each layer, the output value in the hidden layer depends only on a few input values. This helps in training with smaller training sets, and it is less disposed to overfitting.

We used the CNN as an auto-feature extraction along with a classification model, and the Softmax layer. We divided the output as a multi-class. The Softmax layer provides us with the probability of each class as an output in between 0 and 1. The class has a higher probability selected as an output for a particular instance.

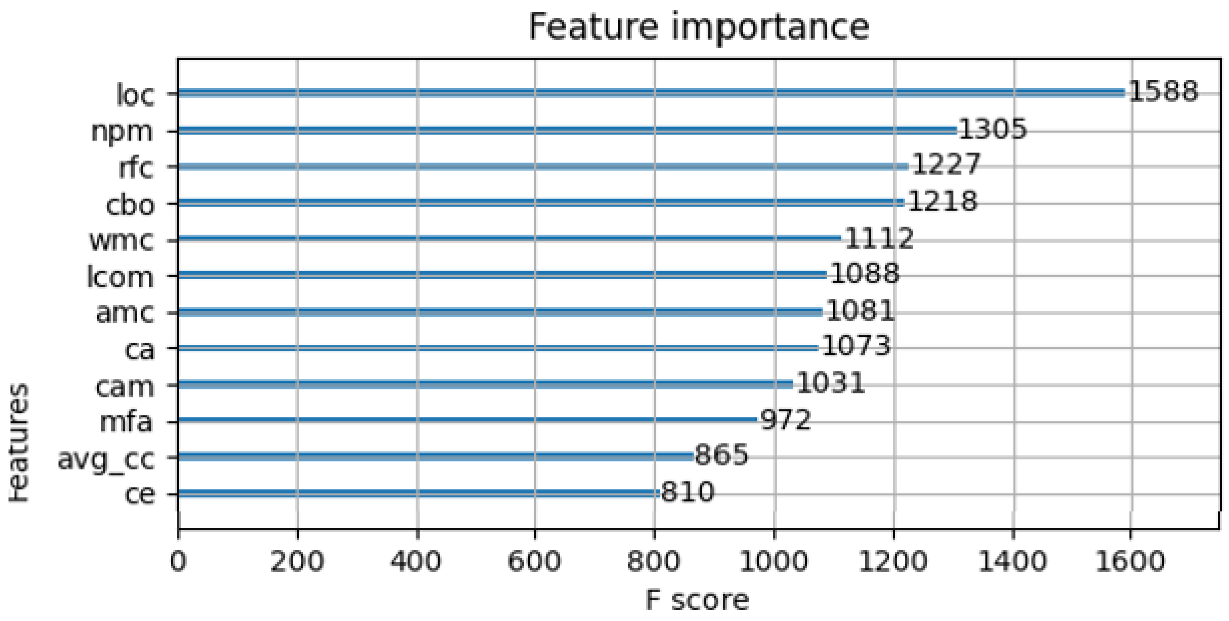


Figure 8. List of 15 important features selected based on their impact on the final output using XGBoost.

Feature extraction is quite a problematic concept concerning the translation of raw data into the inputs that a particular machine learning algorithm desires. It builds valuable information from the raw data features by reformatting, combining, and transforming the primary features into new ones. Until it produces a new dataset that can be used up by the machine learning models to achieve their accuracies, the CNN pooling layer is able to do an auto-feature extraction from the dataset. So, we used the CNN for the auto-feature extraction.

4.6. Model Tuning (Step 6)

We generate our results after we have fine-tuned our model. The error between the last output layer and the actual target is calculated. The calculated error is fine-tuned by adjusting the weights, epochs, and some other parameters to reduce this error.

5. Result and Discussion

In this section, we will answer the research paper's questions and will analyze our results as well. To answer our questions, we have discussed our experimental configuration along with results and analysis. Let us discuss this in detail.

5.1. CNN Architecture Configuration

Table 7 is the experimental configuration of our experiment:

- Batch size: 128.
- Epochs: For projects such as Log4j, Synapse, and Lucene, we set the epoch as 150 and for other projects we set the epoch as 100.
- Optimizer: Adam (learning rate = 0.001).
- Activation function: Softmax.

Table 7. Experimental configuration of CNN architecture.

Model: "Sequential"		
Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 15, 128)	512
leaky_re_lu (LeakyReLU)	(None, 15, 128)	0
max_pooling1d (MaxPooling1D)	(None, 8, 128)	0
dropout (Dropout)	(None, 8, 128)	0
conv1d_1 (Conv1D)	(None, 8, 128)	49,280
leaky_re_lu_1 (LeakyReLU)	(None, 8, 128)	0
max_pooling1d_1 (MaxPooling1D)	(None, 4, 128)	0
Dropout_1 (Dropout)	(None, 4, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 3)	1539

Total Params: 51,331. Trainable Params: 51,331. Non-Trainable Params: 0.

We resolved the issues of noise, a class imbalance, and a distribution gap between the source and the target projects for normalizing the dataset. We treated the output as multi-class instead of binary, as we received strong evidence during the exploratory data analysis that the dataset is multi-class. We applied XGBoost for an optimal feature selection and the CNN for an auto feature extraction along with the classification as well. In this section, we will answer our mentioned research questions.

5.2. Confusion Matrix

A confusion matrix is a tabular way of visualizing the performance of the prediction model. Each entry in a confusion matrix denotes the number of predictions made by the model whether it has classified the classes correctly or incorrectly.

In Figure 9, the horizontal scale mentioned the predicted values of the classes, and the vertical scale mentioned the actual values. Out of 305 zero classes, 240 are predicted as zero class, 48 are predicted as one class, and 17 are predicted as two class. It means that 240 instances are predicted correctly for zero class and 65 are predicted wrongly. In Figure 10, out of 155, 128 are predicted as zero class, 20 are predicted as one class, and 7 are predicted as two class. It means that 128 instances are predicted correctly for zero class and 27 are predicted wrongly and fall under both classes.

RQ1: What is the impact of XGBoost for the optimal feature selection for multi-class in predicting the cross-project defect prediction accuracy of the PROMISE repository?

We evaluated our results by training our classifier by considering the versions of Ant as a source project and all the versions of every dataset as the target projects of the PROMISE repository in terms of the AUC measure. Table 8 effectively shows that our model achieved the maximum higher defect prediction accuracy for the cross-project defect prediction when we used a feature selection using XGBoost before an auto feature extraction using the CNN.

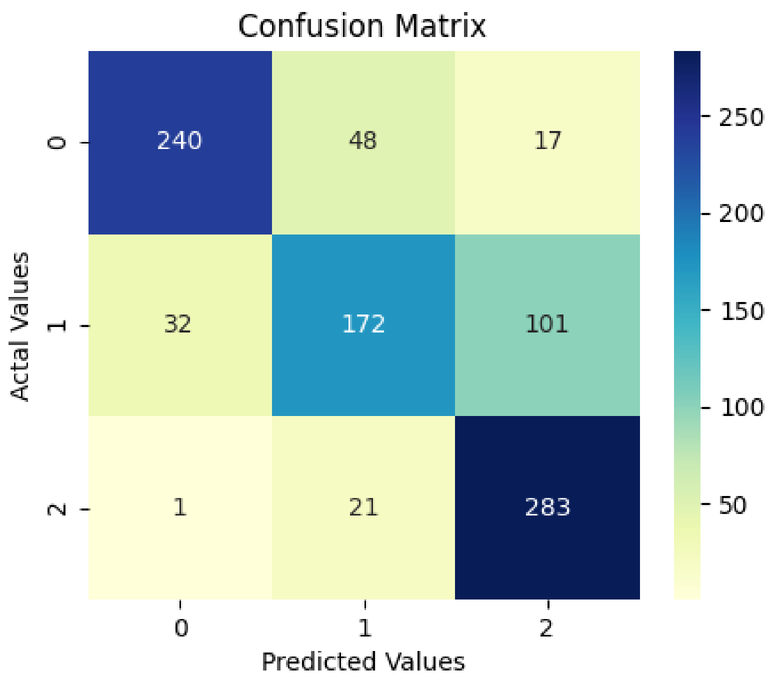


Figure 9. Confusion matrix by considering Ivy-2.0 as target project and ant as source project.

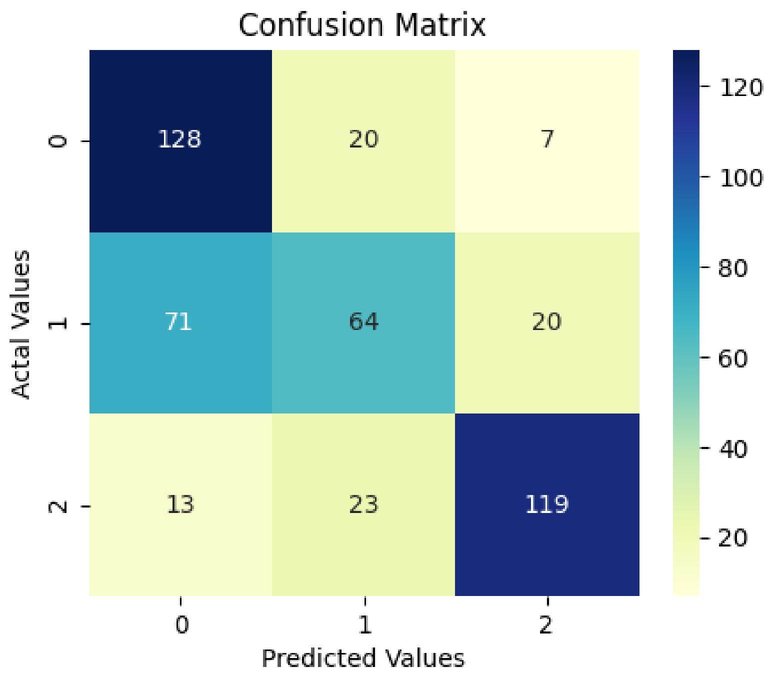


Figure 10. Confusion matrix by considering Synapse-1.1 as target project and Ivy as source project.

In the above Table 8, we witnessed the maximum difference in terms of the accuracy for the Velocity and Xerces to be 12.34%, and a minimum difference of about 2.89% for Lucene, when we compared our result with and without XGBoost in terms of the AUC measure.

We achieved the overall minimum difference for Lucene because the Lucene project is very mature, considering Ant 1.7 as a target project. As the Lucene project has a huge amount of data to train the CTGAN in order to generate a synthetic clone of data, it hence has enough data to train the classifier for a better prediction, which resulted in less difference of the prediction accuracy with and without using XGBoost.

Table 8. Impact on defect prediction accuracy with or without feature selection.

Target	Source	AUC	
		All Features + CNN	**XGBoost + CNN**
Ant 1.7	Ivy	75.67%	82.41%
	Camel	72.04%	81.83%
	Synapse	75.46%	85.95%
	Velocity	49.55%	61.89%
	Lucene	71.58%	73.49%
	Poi	71.05%	76.79%
	Xerces	50.91%	61.61%
	Xalan	75.43%	81.31%
	Log4j	65.65%	68.54%

The high difference for the Velocity and Xerces is due to less amount of multi-class data to train the CTGAN in order to generate a synthetic clone of data and then train the classifier for a better prediction. Both these projects are immature, considering Ant 1.7 as a target project. We needed XGBoost as an important feature selector to train the classifier based on important features having a higher dependency on the output, which resulted in a higher difference of the prediction accuracy with and without selecting optimal features using XGBoost.

Therefore, it is evident from our experimental result that XGBoost selects optimal feature sets, hence the H1 "XGBoost selects optimal feature sets for multi-class in predicting cross-project defect prediction accuracy of PROMISE repository" is accepted.

Now we test the hypothesis of our second research question.

RQ2: What is the impact of the CNN as a classifier in combination with Softmax for multi-class in predicting the cross-project defect prediction accuracy of the PROMISE repository?

In this research question, we are finding the impact of the CNN classifier along with Softmax for the multi-class dataset. The Softmax layer is better suitable for multi-classification because of the range of its output probabilities [11]. The range will be 0 to 1, and the sum of all the probabilities will be equal to one.

The PROMISE repository has 63 output classes. We have the majority of the instances of 0 and 1 classes and a limited number of instances for all the other classes greater than 1 and less than 63. The reason for considering all the classes greater than one is that due to the vast distribution between the number of instances of the source and target projects, the CTGAN is unable to generate synthetic data which covers all instances of the ranges to obtain a higher defect prediction accuracy. The CTGAN first trains the classifier based on the number of instances and then generates the synthetic data on the pattern of the existing classes. So, to overcome this issue and to have greater data for training, we considered all those outliers' classes as class 2.

In the Table 9 below, we achieved higher defect prediction accuracies, highlighted as bold, for those projects in terms of the AUC measure whose dataset ranges are within the ranges of the source projects, such as the existence of a very minimal distribution gap between the source and target projects. The source project has enough number of instances for each class with a minimal distribution gap, so the CTGAN was trained enough and generated enough synthetic clone data for each of the classes to overcome the class distribution problem.

Table 9. Experimental results of each project in terms of AUC.

AUC Measure	**Sources**										
Target	**Ant**	**Camel**	**Jedit**	**Lucene**	**Poi**	**Log4j**	**Velocity**	**Synapse**	**Ivy**	**Xalan**	**Xerces**
Ant-1.3	-	68.46%	63.36%	65.11%	54.01%	57.63%	53.11%	66.62%	**69.04%**	67.84%	***48.33%***
Ant-1.4	-	50%	70.24%	65.65%	59.99%	63.47%	***48.38%***	66.95%	**73.64%**	73.44%	49.65%
Ant-1.5	-	79.84%	75.37%	73.46%	63.46%	58.53%	***52.08%***	79.04%	68.30%	**83.54%**	82.19%
Ant-1.7	-	81.83%	73.15%	73.49%	76.79%	68.54%	61.89%	**85.95%**	82.41%	81.31%	***61.61%***
Camel-1.2	70.81%	-	**71.91%**	71.60%	56.42%	62.32%	67.82%	63.72%	70.10%	62.97%	***53.12%***
Camel-1.4	**74.91%**	-	70.63%	71.37%	59.40%	67.98%	50.73%	65.63%	70.49%	66.41%	***47.63%***
Camel-1.6	**77.56%**	-	66.41%	66.22%	62.71%	68.46%	***51.82%***	68.62%	68.79%	56.72%	58.46%
Ivy-2.0	**88.19%**	77.09%	79.99%	73.72%	59.80%	72.81%	64.07%	75.92%	-	78.15%	***31.67%***
Jedit-3.2	**78.28%**	74.03%	-	67.27%	63.62%	75.91%	65.90%	76.39%	76.93%	58.59	***48.63%***
Jedit-4.0	81.45%	73.66%	-	75.87%	61.58%	66.94%	***52.90%***	72.47%	**81.83%**	77.84%	66.56%
Jedit-4.1	**81.96%**	71.26%	-	73.95%	56.02%	64.58%	62.18%	78.17%	81.47%	78.57%	***55.91%***
Log4j-1.0	**78.23%**	74.87%	72.53%	62.83%	61.31%	-	***53.59%***	63.37%	76.36%	64.23%	60.64%
Log4j-1.1	73.56%	**78.36%**	63.22%	70.57%	71.07%	-	68.18%	60.68%	***57.90%***	65.88%	64.20%
Log4j-1.2	50.18%	***34.23%***	46.58%	**70.01%**	61.38%	-	58.25%	45.73%	47.86%	55.33%	47.28%
Lucene-2.0	71.85%	70.15%	64.93%	-	58.78%	***47.16%***	50.93%	63.40%	**72.52%**	64.27%	53.51%
Lucene-2.2	63.26%	63.48%	61.50%	-	62.54%	53.26%	58.29%	63.58%	58.92%	**65.26%**	***44.10%***
Lucene-2.4	58.57%	65.30%	57.84%	-	60.51%	66.32%	***50%***	55.92%	60.77%	**67.81%**	52.61%
Synapse-1.0	**81.40%**	66.85%	68.53%	62.97%	71.65%	66.17%	***53.89%***	-	80.92%	64.21%	62.74%
Synapse-1.1	79.97%	77.89%	73.71%	57.90%	64.73%	52.91%	***44.43%***	-	**87.65%**	65.81%	53.96%
Synapse-1.2	74.21%	73.39%	71.81%	65.23%	67.41%	52.44%	***48.81%***	-	**75.80%**	70.95%	59.72%
Velocity-1.4	**62.92%**	58.03%	60.01%	62.54%	57.89%	55.16%	-	57.35%	62.20%	64.04%	***50.49%***
Velocity-1.5	61.11%	55.99%	***53.06%***	**73.65%**	73.14%	65.10%	-	53.47%	59.32%	69.45%	63.75%
Velocity-1.6	58.77%	53.08%	58.42%	49.41%	57.19%	***41.49%***	-	**65.58%**	61.96%	51.25%	53.76%
Xalan-2.4	**76.67%**	72.83%	67.93%	67.04%	53.28%	64.38%	***47.94%***	73.04%	76.46%	-	59.27%
Xalan-2.5	**79.40%**	73.48%	71.04%	69.28%	52.75%	57.02%	***46.96%***	76.92%	79.30%	-	56.61%
Xalan-2.6	76.90%	73.15%	69.98%	71.86%	55.43%	71.20%	54.42%	75.18%	**77.49%**	-	***52.98%***
Xerces-1.2	**72.97%**	62.34%	67.41%	49.10%	***34.02%***	40.57%	48.06%	68.16%	69.56%	64.56%	-
Xerces-1.4	61.70%	59.24%	60.33%	53.98%	56.16%	***51.70%***	52.80%	61.09%	**63.58%**	62.36%	-

In Table 9 above, the dashes represent the different versions of the selected target. So, when we select any of the targets, we do not calculate the accuracies for its own different versions because the procedure is used for a within-project defect prediction or a within-version defect prediction, and we are working on this paper as a cross-project defect prediction. Therefore, dashes represent a result of no accuracy for its versions.

We used a combined version of the source instead of considering a single version as the source. To effectively train the classifier, we need too large a number of data, so hence using a combined dataset of different versions consists of different data ranges instead of very few data. We can also be able to generate the required amount of data using the CTGAN for the selected resource, but the CTGAN generates data in the same space as the source with minimal distribution in the dataset. So, in this way, we are unable to receive a better accuracy for the target projects having marginal differences in each feature of the data, as represented in bold italic. By combining all the versions of the source, we have different variations for each feature of the dataset and will be able to obtain a better accuracy for each of the target projects.

Therefore, it is evident from our experimental results that the CNN as a classifier in combination with the Softmax for a multi-class is better for predicting the cross-project defect prediction accuracy in terms of the AUC measure for the PROMISE repository, thus the statement "CNN as classifier in combination with Softmax has an impact for multi-class in predicting cross-project defect prediction accuracy of PROMISE repository" is accepted.

5.3. Research Validation

We validated our results by statistical tests, i.e., Wilcoxon's test. The details are as follows:

5.3.1. Wilcoxon Test

"The Wilcoxon signed ranks test is a nonparametric statistical procedure for comparing two samples that are paired, or related. The parametric equivalent to the Wilcoxon signed ranks test goes by names such as the student's *t*-test, *t*-test for matched pairs, *t*-test for paired samples, or *t*-test for dependent samples".

5.3.2. Hypothesis Assumption

The followings are the assumption for the hypothesis:

H0. *Sample distributions are equal.*

H1. *Sample distributions are not equal.*

5.3.3. Acceptance Criteria

The significance level, based on the information provided, is alpha = 0.05. If the *p* value is greater than the alpha, then H0 is accepted and anything else is rejected.

5.3.4. Test Statistics

Table 10 shows the test results using Wilcoxon's test. We calculated the value of *p* by passing two groups in the Wilcoxon test. We considered Ant as the target project and observation in terms of the accuracy with other source projects such as Ivy, Camel, Synapse, Velocity, Lucene, Poi, Xerces, Xalan, and Log4j. One of the groups of the observation contains the values of measure without applying XGBoost and another observation contains the values of measure after applying XGBoost for research question 1.

Table 10. Test statistics of Wilcoxon h test.

Test Statistics		
	***p*-Value**	**Accepted Hypothesis**
Wilcoxon Test	0.0039	H1

5.3.5. Analysis of Validation test

The result of the validation test as shown in the above figure using Wilcoxon's test is used to validate the result of the experiment we conducted. Our proposed model has less value of *p* compared to the alpha, such as 0.0039 < 0.05, so it is concluded that it rejects the null H0 hypothesis, and our alternative hypothesis is accepted for both of the tests.

6. Threats to Validity

During an empirical study, one should be aware of the potential threats to the validity of the obtained results and the derived conclusions. The potential threats to the validity identified for this study are assessed in three categories, namely: internal, external, and construct validity.

6.1. Hyper Parameter Combination

For a better prediction performance, we tried different parameter combinations of the model. However, it is impossible to experiment with all the possible combinations of hyper parameters.

6.2. Internal Validity

We implement the baselines by carefully following the original papers. These compared related works do not provide the source codes of their works, so we have not worked on the source code of the dataset. We have worked on the static features of the dataset. Although we try our best to guarantee the accuracy of the implementation.

6.3. External Validity

The PROMISE datasets used for the validation are open-source software project data. If our proposed approach is built on the closed software projects developed under different environments, it might produce a better/worse performance.

7. Conclusions

During the EDA of the PROMISE repository, there exists multi-classes which have noise, distribution gap, and class imbalance issues, as evident in Table 3, Figure 8, and Table 5. It is evident from our experimental results that after removing the noise, reducing the distribution gap, and balancing the output classes, by selecting optimal feature sets through XGBoost and then combining Softmax with the CNN classifier, we can improve the cross-project defect prediction accuracy for the multi-class of the PROMISE repository in terms of the AUC measure. In our experimentation for CPDP, we selected all 28 versions of the multi-class projects, in which 11 versions were selected as the source projects and against which we predicted 28 target versions, with the average AUC being 75.57%. We validated our results through the Wilcoxon test. Such a cross-project defect prediction will be fundamental for predicting the defects from software modules and will reduce the time and cost of software developers in finding defects at the initial stages of software development.

We calculated our result based on the fixed 15 optimal features set, which we obtained by applying XGBoost on the Ant project. We applied the same feature set for all of the projects to obtain the defect prediction accuracy. One of the potential ways to improve the defect prediction accuracy of our model is to pass the top 15 features set for each of the projects separately to the CNN model after applying XGBoost. We also fixed the experimental configuration of the CNN model. Another potential way to improve the defect prediction accuracy of our model is to fine-tune the CNN configuration for each project.

Our feature selection and classification are for multi-class data upon a cross-project, i.e., the PROMISE repository. However, such a feature selection and classification may result in a different prediction accuracy for other multi-class data upon a cross-project. Therefore, we cannot generalize our findings for all the multi-class data upon a cross-project.

Author Contributions: Conceptualization, S.N. and R.B.F.; data curation, S.N. and R.B.F.; formal analysis, S.N. and R.B.F.; funding acquisition, S.A. and M.M.; investigation, S.N., R.B.F., S.A. and M.M.; methodology, S.N.; project administration, R.B.F., S.A. and M.M.; resources, S.N.; software, S.N.; supervision, R.B.F.; validation, S.N., R.B.F., S.A. and M.M.; visualization, S.N. and R.B.F.; writing—original draft, S.N.; writing—review and editing, S.N. and R.B.F. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this research–Group No (RG-1436-039).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding this Research–Group No (RG-1436-039).

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

References

1. Wang, A.; Zhang, Y.; Wu, H.; Jiang, K.; Wang, M. Few-Shot Learning Based Balanced Distribution Adaptation for Heterogeneous Defect Prediction. *IEEE Access* **2020**, *8*, 32989–33001. [CrossRef]
2. Menzies, T.; Milton, Z.; Turhan, B.; Cukic, B.; Jiang, Y.; Bener, A. Defect prediction from static code features: Current results, limitations, new approaches. *Autom. Softw. Eng.* **2010**, *17*, 375–407. [CrossRef]

3. Nana, Z.; Kun, Z.; Ying, S.; Wang, X. Software Defect Prediction Based on Non-Linear Manifold Learning and Hybrid Deep Learning Techniques. *Comput. Mater. Contin.* **2020**, *65*, 1467–1486. [CrossRef]
4. Zhu, K.; Zhang, N.; Ying, S.; Wang, X. Within-project and cross-project software defect prediction based on improved transfer naive bayes algorithm. *Comput. Mater. Contin.* **2020**, *63*, 891–910. [CrossRef]
5. Nana, Z.; Kun, Z.; Ying, S.; Wang, X. Software Defect Prediction Based on Stacked Contractive Autoencoder and Multi-objective Optimization. *Comput. Mater. Contin.* **2020**, *65*, 279–308. [CrossRef]
6. Zain, M.Z.; Sakri, S.; Ismail, A.H.N.; Parizi, R.M. Software Defect Prediction Harnessing on Multi 1-Dimensional Convolutional Neural Network Structure. *Comput. Mater. Contin.* **2021**, *71*, 1521–1546. [CrossRef]
7. Pan, C.; Lu, M.; Xu, B.; Gao, H. An Improved CNN Model for Within-Project Software Defect Prediction. *Appl. Sci.* **2019**, *9*, 2138. [CrossRef]
8. Laradji, H.I.; Alshayeb, M.; Ghouti, L. Software defect prediction using ensemble learning on selected features. *Inf. Softw. Technol.* **2015**, *58*, 388–402. [CrossRef]
9. Shepperd, M.; Bowes, D.; Hall, T. The Use of Machine Learning in Software Defect Prediction. *IEEE Trans. Softw. Eng.* **2014**, *40*, 603–616. [CrossRef]
10. Song, Q.; Yuchen, G.; Martin, S. A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction. *IEEE Trans. Softw. Eng.* **2018**, *45*, 1253–1269. [CrossRef]
11. Sun, Y.; Jing, X.Y.; Wu, F.; Li, J.; Xing, D.; Chen, H.; Sun, Y. Adversarial Learning for Cross-Project Semi-Supervised Defect Prediction. *IEEE Access* **2020**, *8*, 32674–32687. [CrossRef]
12. Sun, Y.; Sun, Y.; Qi, J.; Wu, F.; Jing, Y.X.; Xue, Y.; Shen, Z. Unsupervised domain adaptation based on discriminative subspace learning for cross-project defect prediction. *Comput. Mater. Contin.* **2021**, *68*, 3373–3389. [CrossRef]
13. Qiu, S.; Lu, L.; Jiang, S. Multiple Components Weights Model for Cross-Project Defect Prediction. *IET Softw.* **2018**, 12. [CrossRef]
14. Kumar, R.P.; Varma, S.D.G. A Novel Multi-Level Based Cross Defect Prediction Model for Multiple Software Defect Databases. *Int. J. Pure Appl. Math.* **2017**, *117*, 293–301.
15. Khoshgoftaar, M.T.; Gao, K.; Seliya, N. Attribute Selection and Imbalanced Data: Problems in Software Defect Prediction. In Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence, Arras, France, 27–29 October 2010; Volume 1, pp. 137–144. [CrossRef]
16. Dam, K.H.; Pham, T.; Ng, S.W.; Tran, T.; Grundy, J.C.; Ghose, A.K.; Kim, T.; Kim, C. A deep tree-based model for software defect prediction. *arXiv* **2018**, arXiv:abs/1802.00921.
17. Li, J.; He, P.; Zhu, J.; Lyu, R.M. Software Defect Prediction via Convolutional Neural Network. In Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic, 25–29 July 2017. [CrossRef]
18. Qiu, S.; Xu, H.; Deng, J.; Jiang, S.; Lu, L. Transfer Convolutional Neural Network for Cross-Project Defect Prediction. *Appl. Sci.* **2019**, *9*, 2660. [CrossRef]
19. Wang, S.; Liu, T.; Nam, J.; Tan, L. Deep Semantic Feature Learning for Software Defect Prediction. *IEEE Trans. Softw. Eng.* **2020**, *46*, 1267–1293. [CrossRef]
20. Shirabad, S.; Menzies, T.J. Promise Software Engineering Repository. Available online: http://promise.site.uottawa.ca/SERepository/ (accessed on 30 June 2020).
21. Hosseini, S.; Turhan, B.; Mäntyl, 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]
22. Sheng, L.; Lu, L.; Lin, J. An Adversarial Discriminative Convolutional Neural Network for Cross-Project Defect Prediction. *IEEE Access* **2020**, *8*, 55241–55253. [CrossRef]
23. Zhang, J.; Wu, J.; Chen, C.; Zheng, Z.; Lyu, M. A Cross–Version Software Defect Prediction Model with Data Selection. *IEEE Access* **2020**, *8*, 110059–110072. [CrossRef]
24. Chidamber, R.S.; Kemerer, F.C. Towards a metrics suite for object-oriented design. *ACM Sigplan Not.* **1991**, *26*, 197–211. [CrossRef]
25. Martin, R. OO design quality metrics: An analysis of dependencies. *Appl. Sci.* **1994**, *12*, 151–170.
26. Bansiya, J.; Davis, G.C. A Hierarchical Model for Object-Oriented Design Quality Assessment. *IEEE Trans. Softw. Eng.* **2002**, *28*, 4–17. [CrossRef]