Automated Software Engineering (2021) 28:11
https://doi.org/10.1007/s10515-021-00289-8

Discriminating features-based cost-sensitive approach for software defect prediction

Aftab Ali[1] · Naveed Khan[1] · Mamun Abu-Tair[1] · Joost Noppen[2] · Sally McClean[1] · Ian McChesney[1]

Received: 28 December 2020 / Accepted: 24 June 2021

Abstract
Correlated quality metrics extracted from a source code repository can be utilized to design a model to automatically predict defects in a software system. It is obvious that the extracted metrics will result in a highly unbalanced data, since the number of defects in a good quality software system should be far less than the number of normal instances. It is also a fact that the selection of the best discriminating features significantly improves the robustness and accuracy of a prediction model. Therefore, the contribution of this paper is twofold, first it selects the best discriminating features that help in accurately predicting a defect in a software component. Secondly, a cost-sensitive logistic regression and decision tree ensemble-based prediction models are applied to the best discriminating features for precisely predicting a defect in a software component. The proposed models are compared with the most recent schemes in the literature in terms of accuracy, area under the curve, and recall. The models are evaluated using 11 datasets and it is evident from the results and analysis that the performance of the proposed prediction models outperforms the schemes in the literature.

Keywords Software bugs/defects · Machine learning models · Discriminating features · Cost-sensitivity · AUC · Recall

1 Introduction

The development of high performance and efficient software systems is increasing day by day. This efficiency is achieved at the cost of software complexity. Analysing these complex software systems manually is a difficult, tedious and costly process Rathore and Kumar (2017). To overcome this difficulty automatic software defect prediction can play a vital role. Though automated software

✉ Aftab Ali
a.ali@ulster.ac.uk

Extended author information available on the last page of the article

defect prediction of large and complex systems is a challenging task, it can be accomplished by utilising correlated quality metrics.

The cost involved in finding the modules or components containing defects is also a challenging problem for the software development industry. A Cambridge University study shows that the cost of software defects has increased to $312 billion per annum globally Brady (2013). The reason behind the cost increase is that developers spent most of their time on finding and fixing defects. The developer's goal is to release defect free software to the end user. Unfortunately, software defects are inevitable; for example the US Department of Defence is spending over four billion dollars for software failures per year Dick et al. (2004).

Techniques of software testing are typically used to reduce defects and ensure high quality systems However, such testing requires some tedious and exhaustive test cases to be executed, and this makes the process quite expensive, especially since the defect removal effectiveness of traditional testing activities can be very low Ebert and Jones (2009) Arar and Ayan (2015) Kassab et al. (2017) Ammann and Offutt (2016) Jorgensen (2018) Huda et al. (2017). According to NIST xxx (2002) and Eckardt et al. (2014), finding defects early in the development process greatly lowers the average cost of defects. Similarly, it is also widely recognised that finding defects early in the development process greatly lowers the cost of removal Sommerville (2004). Moreover, inadequate software testing is causing $3.3 billion to U.S. software developers and users in the financial services sector Eckardt et al. (2014). According to a study software defects cost U.S. industry $60 billion a year Tassey (1996). This supports our case that accurate and automatic software defect prediction should be an important part of the software development process.

In Gyimothy et al. (2005), the authors used a p-value based ranking of metrics for finding contributing features for defect prediction using logistic regression. The authors performed experiments on the open source web and e-mail suite Mozilla version 1.7. The authors examined a total of eight performance metrics, where only one metric was found to be non-contributing. In another paper, Malhotra (2015) reviewed such approaches and found that logistic regression performed poorly, in terms of prediction, when compared with machine learning approaches. However, logistic regression provides good explainability and when used in combination with the best feature selection scheme and cost-sensitive function its performance improves significantly in predicting defects and quantifying this.

In our experiments we use 11 datasets from the PROMISE Shirabad and Menzies (2005) software engineering repository as can be seen in Table 1. It is evident from Table 1, that most of the datasets are highly skewed (i.e. the number of defect instances are very small compared to the normal instances). Therefore, our focus in this work is to precisely predict the defects in the highly skewed data. For this our current approach utilised the cost-sensitive function to handle the data skewness and to improve prediction performance. In this paper, we propose best features-based cost-sensitive logistic regression (CLR) and decision trees ensemble (CDTE) models to predict software defects. The proposed approaches first find best discriminating attributes (features) for defect prediction by using analysis of variance (ANOVA) F-value. Once those features are identified in the next step, CLR

Table 1 Datasets description and level of class imbalance

Dataset description			
Dataset	Defects	Normal	Defects percentage
kc1	326	1783	15.46
kc2	107	415	20.50
pc1	77	1032	6.94
pc3	160	1403	10.24
pc4	178	1280	12.21
Mozilla4	5108	10437	32.86
mc1	68	9398	0.72
mc2	52	109	32.30
cm1	50	448	10.04
jm1	2104	8776	19.34
jEdit_4.2_4.3	204	165	55.28

and CDTE prediction models are applied to the selected attributes to predict those instances which contain defects.

In our prior work Ali et al. (2019), we used the p-value to find the discriminating features for software defects detection. We observed that p-value do not directly quantify any useful measures of model quality and hence resulted in poor detection of actual software defects (i.e. very low recall). Therefore, we now consider the F-value in combination with the p-value to select the best discriminating features along with the incorporation of cost-sensitivity to improve the detection of actual defects (i.e. recall).

The rest of the paper is organised as follows: Sect. 2 presents motivation and scope of the work while Sect. 3 elaborates the defect prediction related work. Section 4 describes the proposed cost-sensitive defect prediction using best features-based logistic regression and decision trees ensemble. Results and discussions are covered in Sect. 5, while Sect. 6 presents the limitations and future directions. Finally, Sect. 7 concludes the work.

2 Motivation and scope

2.1 Motivation

Software developers face challenges to understand increasingly complex software and identify and resolve defects efficiently. This is critically important but often very hard to achieve. The analysis of software development processes established the need to provide support for early identification of high-risk areas to help prioritise developer effort and improve efficiency and quality.

With the increasing size of software, it is of key importance for organisations that developers can easily explore and prioritise work. The aim of the defect detection process is to guide the developers to the most critical and fragile parts of software system during development. Effective software defect prediction will increase the performance of the application being examined and will reduce the cost of software development. Moreover, with defect detection and resolution this research will optimize resource use and prevent costly mistakes making it into production systems.

2.2 Scope

The scope of the work is quite broad in a sense that good quality version controlling and error reporting will help in the calculation of the qualitative metrics used in this research. Once the qualitative metrics are being extracted from the source code, then the next step is to apply the models on those metrics. The scope of this work is limited to the labelled qualitative metrics extracted from the source code repositories.

3 Related work

A number of machine learning algorithms have been described in the literature to address software fault prediction problems. In Malhotra (2015), the authors have performed a systematic review of machine learning techniques used for software defect prediction. Moreover, a detailed analysis about the performance of statistical and machine learning techniques has been carried out to identify the strength and weaknesses of these techniques used for software defect prediction. Generally, there is a trade-off achieving higher defect detection rate and higher accuracy rate, therefore, the machine learning model is considered optimum? if it has attained both higher defect detection rate and higher accuracy rate. Therefore, the random forest approach has been used in Guo et al. (2004) for software defect prediction as it has previously achieved a higher defect detection rate and higher accuracy as compared to other benchmark techniques such as discriminant analysis and decision tree algorithm.

The authors in Catal and Diri (2009), have used correlation-based feature selection (cfs) technique to extract discriminant features from the public NASA dataset. The discriminant features are further analysed using artificial immune system algorithms for software fault prediction. The performance of the machine learning algorithm has improved using feature reduction technique for software fault prediction.

Likewise, in Osman et al. (2017), the effect of embedded feature selection algorithms has been analysed to improve the performance of the predictive model for defect prediction. The authors have used three regularization methods Ridge, Lasso and ElasticNet on Poisson and linear Regression prediction algorithm. The results show that the proposed technique improves the performance by minimizing the root mean squared error up to 50% and also improves the stability of the prediction model.

Moreover, Huda et al. (2017) have proposed a hybrid wrapper-filter approach for metric selection. The Support Vector Machine (SVM) hybrid heuristics and Artificial Neural Network (ANN) hybrid heuristics have been used to find discriminant metrics from software defect data. Further, both hybrid models SVM and ANN were copulated with a maximum relevance (MR) filter to extract more significant features that achieved high prediction accuracy in software defect prediction. Recently, Son et al. (2019) conducted an empirical study on defect prediction in software (DeP) that covers the prediction and classification techniques based on defect severity and security-related defects for software. The authors concluded that an ideal DeP should have the capability of accurately identifying the defects, and should classify different types of defects on the basis of severity, security-related defects etc.

Arshad et al. (2018) proposed a semi supervised Deep Fuzzy C-Mean clustering approach for software defect prediction. The proposed approach analysed supervised and unsupervised data simultaneously to extract useful information from the dataset. The results of the proposed approach are compared with the semi supervised techniques used for software fault prediction. Their approach has outperformed benchmark approaches in achieving high AUC and F-measure scores. Arasteh (2018) proposed a fault prediction method using Neural Network and Naïve Bayes algorithms. The author compares the proposed approach with three different defect prediction models including Support Vector Machine (SVM), Artificial Neural Nketwork (ANN) and Naïve Bayes. The results have shown that the proposed algorithm outperforms other prediction models in terms of achieving high accuracy and precision for software fault prediction.

Most of the schemes in the literature suffer from the class imbalance problem, and hence result in poor prediction of the defects for complex software systems Ali et al. (2019). As can be seen in Table 2, that most of the schemes in literature focuses on the performance metrics like, accuracy, AUC for the evaluation. However, the use of these metrics in imbalanced classes can lead to sub-optimal classification models and might yield misleading results due to insensitivity towards skewness in data Branco et al. (2015). For example, a no skill classification model might produce high accuracy (or low error) by only predicting the majority class.

As for a good quality software system the probability of getting lesser number of defects containing instances is very high, resulting in a highly imbalance data, causing the prediction algorithm to mispredict the actual defects. Therefore, in this paper we propose best features-based cost-sensitive logistic regression (CLR) and decision trees ensemble (CDTE) models to predict software defects. The

Table 2 A literature review and comparison

	Accuracy	AUC	Precision	Recall	F-measure	MAE	RMSE
Huda et al. (2017)	✓	✓					
Ali et al. (2019)	✓	✓					
Arar and Ayan (2015)	✓	✓					
Esteves et al. (2020)	✓	✓			✓		
Proposed models	✓	✓	✓	✓	✓	✓	✓

cost-sensitive learning directs the classification towards optimising the desired performance criteria, namely focusing on optimally detecting the defects.

4 Cost-sensitive defect prediction using best features-based logistic regression and decision trees ensemble

Preprocessing of data to extract useful information (also referred as feature extraction) is one of the most crucial steps in classification and prediction tasks Pendharkar (2005). The accuracy of classification and prediction is strongly dependent on the extraction of relevant features from the raw data Aparna and Paul (2016).

Analysis of variance (ANOVA) Freedman (2009) is a statistical measure used to check if the means of two or more features are significantly different from each other. For the feature selection process, we use the ANOVA F-value (i.e. an estimation of the degree of linearity between the input metric and the output metric). The ANOVA F-value is sequentially applied to all features and it will then select those features which are more discriminating according to classes. The ANOVA F-value-based feature selection process will remove redundant and irrelevant features from the data set as often as possible and will select an optimal feature subset for precise prediction of defects. Significant or best features have an impact on the outcome of any classification algorithm and can increase the performance efficiency even with a reduced set of training data. Significant/best feature analysis improves scalability, and processing efficiency for the training and testing of the classifier.

Once the best features are selected using the ANOVA F-value based feature selection process then the next step is to pass those features to a classification algorithm for prediction. In this work the classification and prediction schemes are based on logistic regression and decision tree ensemble (J48 ensemble). The problem with using logistic regression and decision trees ensemble is that it suffers from the class imbalance problem. It is also obvious that in any typical software system the number of instances containing defects will be lower than the normal instances i.e., the probability of getting highly skewed data is very high. To remedy this situation, a MetaCost-based Domingos (1999) logistic regression and decision tree ensemble are used. A simple logistic regression is given in the following Equation 1:

$$log\left(\frac{\pi}{1-\pi}\right) = a + b_1 x_1 + \ldots + b_n x_n \tag{1}$$

where $(\pi/(1-\pi)$ is the odds ratio; $a, b_1, \ldots b_n$ are the coefficients of the regression model esti-mated (learned) from the repository data; and $x_1, \ldots x_n$ are the features which we extract from the repository data e.g. loc, cyclomatic complexity etc.

Logistic regression is used for predictive analysis of data to identify the relationship be-tween a dependent binary variable and one or more independent variables by estimating the probabilities using a logistic function Le Cessie and Van Houwelingen (1992). The reason for choosing logistic regression is that it is quite extensible and explainable Catal (2011) Sunil et al. (2018) and it is also fairly simple in terms of implementation. While utilising the advantage of logistic regression simplicity and expandability, we also used decision trees ensemble (CDTE) to add some more diversity to the defect prediction process.

A simple and generic diagrammatic representation of the proposed cost-sensitive discriminating/best features-based classification model is presented in Fig. 1, where the costs are applied during the training phase to make the model learn the associated costs. The models are first trained with a cost function to minimize the misclassification errors (i.e. in our case defects). In order to achieve minimal defects mispredictions, the models are trained with higher costs in the defective class.

This cost assigning and learning model is applied on the logistic regression and decision tree ensemble to make these models more appropriate for defect prediction and classification or in other words to reduce the cost of defects misclassifications. Costs learning process is common for both logistic regression and decision tree ensemble.

5 Experiments and results

This section provides the details of datasets used in the experiments, the tools and experimental setup and discussion on the experiments and results. In order to test the models for various parameters like, time taken for generating the results (i.e. models execution time), the simulation is performed in Ubuntu. The models are implemented in Python using Jupyter notebook on Ubuntu virtual machine (VM). The VM has 160 GB storage and 16 GB memory with a single processor. The host machine is running Windows 10 and has a 16 GB physical memory with a Core i5 CPU (1.70GHz 1.90 GHz).

The performance of CLR and CDTE models are assessed using 10-fold cross validation, where the datasets are partitioned into a training set to train the model and a testing set to test the model. In 10-fold cross-validation, the data is divided into 10 equally sized parts, called folds, where the 9 folds are utilized for training the model and one fold is kept for validation. The trained model is then applied to predict the target variable in the testing data and the process is repeated 10 times, with the performance of each model in predicting the set being hold. The performance is measured by calculating metrics such as accuracy, area under the curve (AUC), recall, precision, F-measure, Mean Absolute Error (MAE), and root mean square

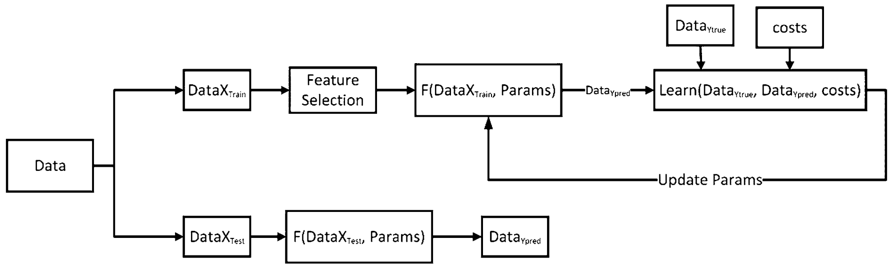

Fig. 1 Cost-sensitive classification model

error (RMSE). The advantage of this approach is that the input dataset is used for both training and testing, and each observation is used for testing exactly once.

5.1 Datasets description

In our experiments 11 datasets from the PROMISE Shirabad and Menzies (2005) software engineering repository are utilised as can be seen in Table 1. The feature selection process removes redundant and irrelevant features from the dataset as often as possible and selects an optimal feature subset for precise prediction of defects. As stated earlier, significant or best features have an impact on the outcome of any classification algorithm and can increase the performance efficiency even with a reduced set of training data. In our experiments the feature set is reduced by 60% in some cases (for example in the pc3 dataset, the total features are 37 and the reduced set contains only 15 features). On average the features are reduced by approximately 50% for all the 11 datasets to improve scalability and reduce the processing overhead of the classifier.

As a case study, only one dataset (i.e. pc3) is selected to explain and dry run the feature selection process, but this same process is repeated for all the datasets. Below the process of best discriminating feature selection is explained with reference to the pc3 dataset. Table 3 provides an F-value-based ranked list of pc3 dataset features. The feature selection process selects the best nine features out of the 37 features as given in Table 4.

The higher F-value of a feature increases the chances for that particular feature to contribute to the defect prediction process. Though F-statistic is fairly random, thus, there is a high likelihood that (some) bad features also might have a high F-statistic, especially given the number of tests conducted. Apart from this issue, there is still a good reason to rank the features on the F-statistic e.g., because there is at least a strong likelihood to only drop bad features, even though this approach may also keep bad features. Different number of features were used in the experiments for every dataset to tune the performance of the predictor for best results (i.e. in case of pc3 the optimal results were obtained by using 9 features).

5.2 Analysis and discussion

The two important parameters False Positive (FP) and False Negative (FN) should be considered for a cost-effective prediction process Taylor (2015). FP represents an instance that does not contain any defects but has been identified as a defect by the predictor, whereas FN refers to the undetected defects by the predictor. Now, consider the confusion matrixes in Fig. 2, Logistic regression has a very low FP (i.e. 2%), but the FN (i.e. 83%) is very high. Similarly, the decision tree ensemble has almost the same results, this means that the predictors mis-predict the defects and consider them as normal instances.

We can conclude that the imbalance between the class frequencies in the test data appeared to be causing logistic regression and decision trees ensemble algorithms to ignore the defects and return a default prediction of success Taylor

Table 3 Sorted attribute list of pc3 dataset

S. No	Attribute	F-value	S. No	Attribute	F-value
1	LOC_BLANK	196.97	20	CYCLOMATIC_COMPLEXITY	9.66
2	LOC_COMMENTS	125.15	21	DESIGN_DENSITY	9.59
3	PERCENT_COMMENTS	95.47	22	NUM_OPERATORS	9.04
4	LOC_CODE_AND_COMMENT	81.94	23	BRANCH_COUNT	9.00
5	NUMBER_OF_LINES	64.24	24	HALSTEAD_LENGTH	8.91
6	NUM_UNIQUE_OPERATORS	48.55	25	NUM_OPERANDS	8.60
7	MAINTENANCE_SEVERITY	48.08	26	MULTIPLE_CONDITION_ COUNT	8.45
8	ALL_PAIRS	47.91	27	MODIFIED_CONDITION_ COUNT	8.18
9	NUM_UNIQUE_OPERANDS	35.52	28	HALSTEAD_DIFFICULTY 7.54	
10	DECISION_DENSITY	31.96	29	CONDITION_COUNT	6.86
11	NORMALIZED_CYLOMATIC_ COMPLEXITY	22.80	30	DECISION_COUNT	5.50
12	LOC_TOTAL	21.53	31	HALSTEAD_VOLUME	4.18
13	HALSTEAD_CONTENT	20.98	32	HALSTEAD_ERROR_EST	4.16
14	CYCLOMATIC_DENSITY	16.54	33	PARAMETER_COUNT	3.39
15	HALSTEAD_LEVEL	16.39	34	ESSENTIAL_COMPLEXITY	1.43
16	LOC_EXECUTABLE	15.97	35	ESSENTIAL_DENSITY	0.84
17	NODE_COUNT	10.73	36	HALSTEAD_PROG_TIME	0.00
18	EDGE_COUNT	10.55	37	HALSTEAD_EFFORT	0.00
19	DESIGN_COMPLEXITY	10.17			

Table 4 Best selected features using F-value

S. No	Attribute	F-value
1	LOC_BLANK	196.97
2	LOC_COMMENTS	125.15
3	PERCENT_COMMENTS	95.47
4	LOC_CODE_AND_COMMENT	81.94
5	NUMBER_OF_LINES	64.24
6	NUM_UNIQUE_OPERATORS	48.55
7	MAINTENANCE_SEVERITY	48.08
8	CALL_PAIRS	47.91
9	NUM_UNIQUE_OPERANDS	35.52

(2015). This is not acceptable especially in a cost-effective defect prediction environment because the defects in a software system can cause severe problems in terms of results and expense.

Moreover, to get a complete insight consider Table 5, which shows the True Positives Rate (TPR)/Recall, and AUC for both logistic regression and decision

Table 5 Logistic regression and Decision Tree Ensemble performance without cost sensitive function

Scheme	Dataset	TP Rate/Recall %	AUC %
Logistic Regression	CM1	12.2	80
	JM1	11.6	71.4
Decision Tree Ensemble	CM1	2	61.9
	JM1	14.2	72.4

trees ensemble. It can be observed that the TPR/Recall of the instances containing defects is very low.

To remedy this situation, a MetaCost approach Domingos (1999) along with best feature selection scheme is utilised, which can extend an existing predictor with cost sensitivity. MetaCost creates several re-samples of the training data, uses the predictor to build an ensemble of models trained on those re-samples, then relabels each instance in the original training data with the highest probability class resulting from applying the ensemble set of classiers.

MetaCost allows the Logistic regression and decision tree ensemble to consider higher cost of not predicting a defect, which is an essential feature of the software testing process. In Table 6 and Table 7 the TPR/Recall is now comparatively very high. This is because the schemes are now predicting the defects more accurately.

The AUC is a measure of predictor performance, which shows the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. The AUC value lies between 0.5 to 1, where 0.5 denotes a bad classifier and 1 denotes an excellent classifier. In our case for all the datasets on average the value of 77.55 indicates quite good performance as can be seen in Table 6. Here it is also worth mentioning that the AUC for logistic regression and Decision Trees Ensemble without MetaCost function is almost the same as with the inclusion of MetaCost function, which indicates that the performance is not affected by incorporating the cost-sensitive functionality.

Another important measure of predictor performance is recall, which depicts that how good is the predictor in predicting the actual defects as given in Equation 2. In our experiments we achieved higher values for recall, which indicates that the predictor does a good job in discriminating between the modules containing defects and the ones without a defect which comprise our target variable as presented in Table 6 and Table 7.

$$recall = \frac{TP}{TP + FN} \tag{2}$$

Now, by utilizing the MetaCost function with the best features-based logistic regression and decision trees ensemble approach the experiments were performed, and the analysis has been presented in Table 6 and 7. The accuracy is somehow decreased as compared to the logistic regression and decision trees schemes without cost-sensitive function, while the AUC is almost the same. But, if we look at the confusion matrix given in Fig. 2, Fig. 3, the prediction of defects has been improved

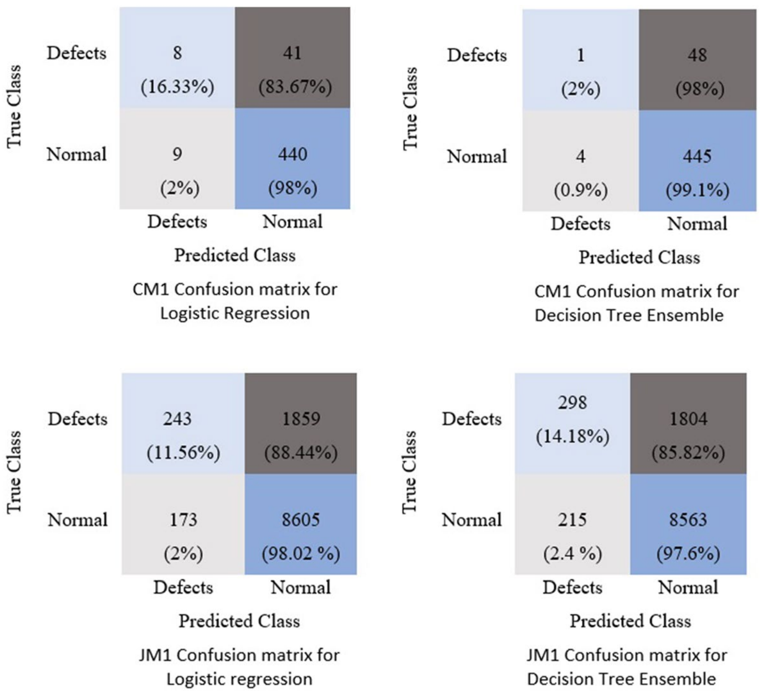

Fig. 2 Confusion matrix without cost-sensitive function

Table 6 Best features-based cost-sensitive Logistic regression performance

Scheme	Dataset	Accuracy %	Recall %	AUC %
Cost-sensitive Logistic Regression	kc1	73	85	81
	kc2	80	85	85
	pc1	75	88	85
	pc3	80	75	70
	pc4	75	91	78
	Mozilla4	75	92	80
	mc1	97	68	80
	mc2	75	70	72
	cm1	78	80	78
	Average	80.27	77.91	77.55

significantly (i.e. from 16% and 11% to 79.6% and 78.6% for the CM1 and JM1 datasets for logistic regression, and from 2% and 14% to 77.6% and 78.7% for the CM1 and JM1 datasets for ensemble bagged trees, respectively).

The models execution time is very low which clearly demonstrates its efficiency. It is observed that the assignment of higher cost values during the experiments due to class imbalance causes the increase in the execution time. This

Table 7 Best features-based cost-sensitive decision tree ensemble performance

Scheme	Dataset	Accuracy %	Recall %	AUC %
Cost-sensitive decision tree ensemble	kc1	69	80	83
	kc2	75	83	85
	pc1	88	61	87
	pc3	80	68	82
	pc4	86	87	91
	Mozilla4	93	93	95
	mc1	98	67	95
	mc2	65	60	68
	cm1	70	78	80
	jm1	65	79	73
	jEdit	65	79	69
	Average	77.64	75.91	82.55

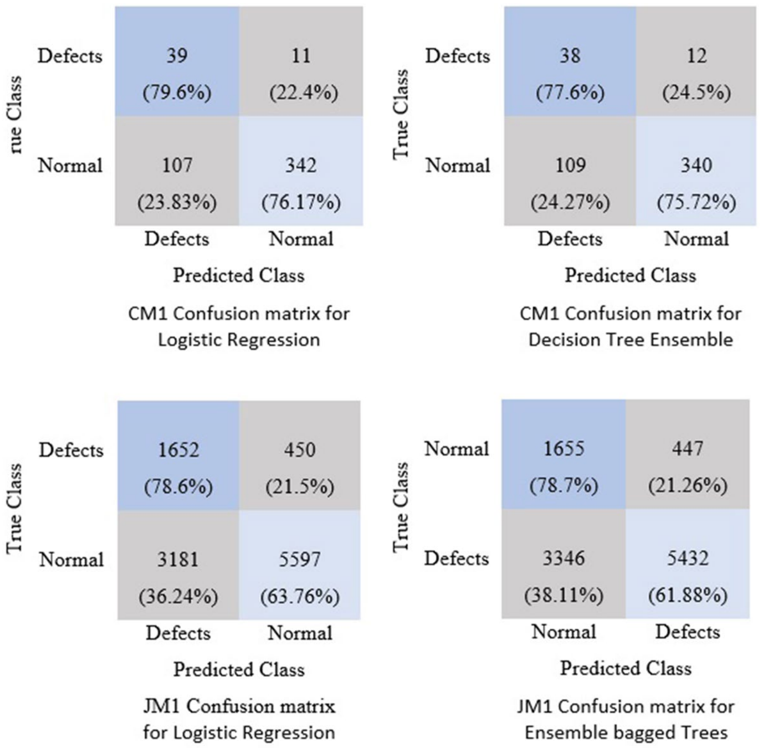

Fig. 3 Confusion matrix with best features-based cost-sensitive logistic regression and decision tree ensemble

increase can be seen for CDTE model in case of kc1 dataset and for CLR model in the case of mc1 dataset where the execution time is 0.56 seconds for both the models. The models are also evaluated using the parameters like Mean Absolute Error (MAE), root mean square error (RMSE), precision and F-measure. MAE measures the average magnitude of the errors in the prediction set. In other words, MAE calculates the difference between the predicted and the actual values. It is evident from Table 8 that the models have low MAE values, which indicates that the models performed better in terms of classifying the defects and normal instances. Similarly, the models also have a very low RMSE, which calculates the average squared errors of the set of predictions.

The models performance is also evaluated in terms of precision and F-measure. Due to the class imbalance in all the datasets used in the experiments, the aim of this research is to identify the defects instances more accurately. Therefore, the weighted averages of both precision and F-measure are calculated, which encompass predictions of majority and minority groups in the data. It can be seen in Table 8, that both the models performed very well in terms of precision and F-measure. This clearly demonstrates that the models accurately classified the defects and normal instances.

Table 8 Execution time and model summary

Dataset	Model	E-Time/seconds	Precision %	F-Measure %	RMSE	MEA
kc1	CDTE	0.56	77.2	79.4	0.45	0.25
	CLR	0.15	83.9	77.2	0.4	0.31
kc2	CDTE	0.13	82.1	78	0.44	0.32
	CLR	0.05	82.9	81.6	0.39	0.31
pc1	CDTE	0.04	92.2	87.6	0.26	0.1
	CLR	0.05	84.3	82.1	0.38	0.27
pc3	CDTE	0.07	88.3	80.5	0.37	0.19
	CLR	0.11	88.2	83.3	0.33	0.23
pc4	CDTE	0.06	90.9	84.9	0.29	0.13
	CLR	0.11	89.1	80.5	0.34	0.19
Mozilla4	CDTE	0.28	94.5	93.8	0.2	0.07
	CLR	0.19	81.8	78.5	0.38	0.26
mc1	CDTE	0.15	98.4	98.7	0.08	0.01
	CLR	0.56	93.4	95.3	0.1	0.03
mc2	CDTE	0.03	60.1	64.3	0.46	0.33
	CLR	0.03	74.1	73.6	0.45	0.29
cm1	CDTE	0.08	87.4	78.5	0.41	0.2
	CLR	0.04	88.3	82.8	0.29	0.37
jm1	CDTE	0.51	77.3	68.7	0.44	0.29
	CLR	0.37	77.9	67.8	0.36	0.32
jEdit	CDTE	0.03	63.2	60.2	0.47	0.38
	CLR	0.04	65.3	75.7	0.43	0.39

5.3 Comparative analysis

In Arar and Ayan (2015), the authors used a cost-sensitive Artificial Neural Network (ANN) model to predict the defective instances in five datasets (i.e. kc1, kc2, pc1, cm1, jm1) from PROMISE Shirabad and Menzies (2005). The proposed best features-based cost-sensitive logistic regression (CLR) and decision trees ensemble (DTE) schemes are compared with Arar and Ayan (2015) in terms of accuracy, AUC and recall.

It is evident from Fig. 4 that the proposed best features-based cost-sensitive logistic regression (CLR) and decision trees ensemble (DTE) schemes have higher accuracy when compared with the scheme in Arar and Ayan (2015).

The proposed best features-based cost-sensitive logistic regression (CLR) and decision trees ensemble (DTE) schemes are also compared with the scheme in Arar and Ayan (2015) in terms of AUC. The proposed schemes has shown good improvement over the existing cost-sensitive approach in Arar and Ayan (2015) as can be seen in Fig. 5.

Moreover, the schemes are also compared in terms of recall, which shows how good the schemes are in predicting actual defects. In Fig. 6, it can be observed that the proposed schemes have outperformed the scheme in Arar and Ayan (2015) in terms of recall, which demonstrates that comparatively the proposed scheme has a very good prediction capability. Due to high class imbalance the performance of CDTE scheme is comparatively poor for only the pc1 dataset.

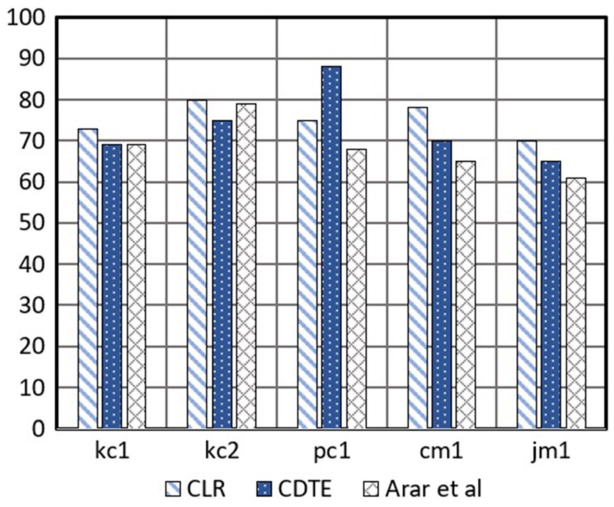

Fig. 4 Comparative analysis in terms of accuracy

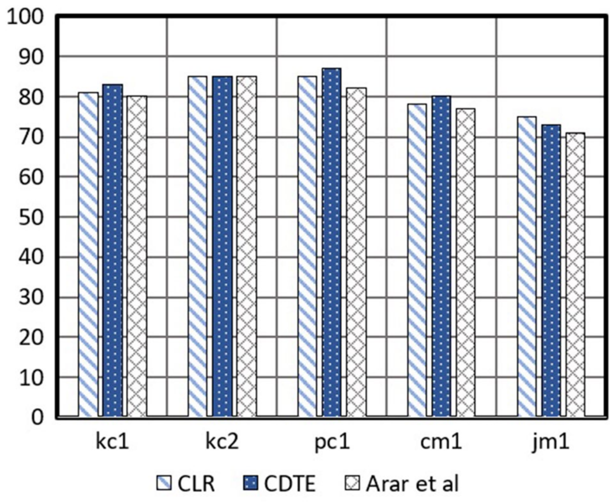

Fig. 5 Comparative analysis in terms of AUC

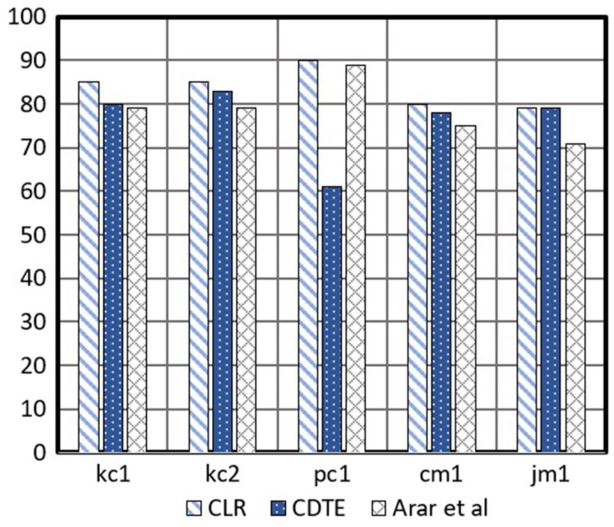

Fig. 6 Comparative analysis in terms of recall

6 Limitations and future directions

The following section describes some of the limitations and future directions to extend our work:

– The current cost-sensitive classification model only aims to pursuit a classifier with the fixed misclassification cost throughout the experiments to minimise the misclassification and maximize the accuracy. However, it does not consider the variable cost for the misclassification. In the future, the work will focus on the variable/dynamic cost assignment for the classification algorithm, rather than using a predefined and fixed cost matrix.

- The accuracy of the classification model is based on the availability of the good quality training data. Hence, there is a need for recording of defective instances in an accurate manner. This can be done by using appropriate commits during the software development process.
- The availability of limited training data may cause problems for the defect detection model. Hence, there is a need for a model which will keep the developer in the loop for the retraining of the model. This will incorporate the experience of the developer in training of the model.
- The performance of the models varies for different datasets. There are a large number of different classification models and there are no “horses for courses” meaning that we cannot say in advance which classifiers might be successful for a given problem or setup. We will therefore use a solution where different classification models could be combined in a way that allows the data to drive the way in which they are combined.

7 Conclusion

Late discovery of defects in software systems is very costly, hence, there is a need for more accurate and efficient automated software defect prediction. There are numerous software quality metrics available in the literature which can be utilised to design a defect prediction system. These quality metrics include traditional code metrics such as McCabe’s complexity measures and Hallstead’s Software Science, object oriented metrics of which the Chidamber and Kemerer suite (1994) is the most widely cited. A challenge for the practitioner is to know which metrics are most valuable in improving the accuracy and efficiency of defect prediction and how they should be used.

In this paper, our focus is to precisely predict the defects in the highly skewed data by selecting the best discriminating features. Once the best discriminating features are selected, we then apply the two cost-sensitive prediction schemes (i.e. CLR and DTE) on the selected metrics for accurate prediction of defects. The performances of the proposed schemes are compared with recent schemes, and it is evident from the results that the proposed schemes perform better in terms of accuracy, AUC, and recall.

Acknowledgements This research is supported by the BTIIC (BT Ireland Innovation Centre) project, funded by BT and Invest Northern Ireland.

Declarations

Conflict of interests The author(s) declare(s) that there is no conflict of interest regarding the publication of this manuscript.

References

Ali, A., Abu-Tair, M., Noppen, Joost., McClean, Sally., Lin, Zhiwei., McChesney, Ian.: Contributing features-based schemes for software defect prediction. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 350–361. Springer (2019)

Ammann, Paul, Offutt, Jeff: Introduction to software testing. Cambridge University Press, Cambridge (2016)

Aparna, UR., Paul, S.: Feature selection and extraction in data mining. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–3. IEEE (2016)

Arar, Ömer F., Ayan, K.: Software defect prediction using cost-sensitive neural network. Appl. Soft Comput. **33**, 263–277 (2015)

Arasteh, B.: Software fault-prediction using combination of neural network and naive bayes algorithm. J. Netw. Technol. **9**(3), 95 (2018)

Arshad, Ali, Riaz, Saman, Jiao, Licheng, Murthy, Aparna: Semi-supervised deep fuzzy c-mean clustering for software fault prediction. IEEE Access **6**, 25675–25685 (2018)

Brady, F.: Cambridge university study states software bugs cost economy \$312 billion per year. *Cambridge University* (2013)

Branco, P., Torgo, L., Ribeiro, R.: A survey of predictive modelling under imbalanced distributions. arXiv preprint arXiv:1505.01658 (2015)

Catal, Cagatay: Software fault prediction: A literature review and current trends. Expert Syst. Appl. **38**(4), 4626–4636 (2011)

Catal, Cagatay, Diri, Banu: Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Inf. Sci. **179**(8), 1040–1058 (2009)

Chidamber, Shyam R., Kemerer, Chris F.: A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, 20(6):476–493 (1994)

Dick, Scott, Meeks, Aleksandra, Last, Mark, Bunke, Horst, Kandel, Abraham: Data mining in software metrics databases. Fuzzy Sets Syst. **145**(1), 81–110 (2004)

Domingos, P.: Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164 (1999)

Ebert, Christof, Jones, Capers: Embedded software: Facts, figures, and future. Computer **42**(4), 42–52 (2009)

Eckardt, James R., Davis, Timothy L., Stern, Richard A., Wong, Cindy S., Marymee, Richard K., Bedjanian, Arde L.: The path to software cost control. *Defense Acquisit. Technol. Logist.*, pages 23–27 (2014)

Esteves, Geanderson, Figueiredo, Eduardo, Veloso, Adriano, Viggiato, Markos, Ziviani, Nivio: Understanding machine learning software defect predictions. Autom. Softw. Eng. **27**(3), 369–392 (2020)

Freedman, David A.: *Statistical models: theory and practice*. cambridge university press (2009)

Guo, L., Ma, Y., Cukic, B., Singh, H.: Robust prediction of fault-proneness by random forests. In *15th international symposium on software reliability engineering*, pages 417–428. IEEE (2004)

Gyimothy, Tibor, Ferenc, Rudolf, Siket, Istvan: Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Trans. Softw. Eng. **31**(10), 897–910 (2005)

Huda, S., Alyahya, S., Ali, Md M., Ahmad, S., Abawajy, J., Al-Dossari, H., Yearwood, J.: A framework for software defect prediction and metric selection. *IEEE access*, 6:2844–2858 (2017)

Jorgensen, Paul C.: *Software testing: a craftsman's approach*. CRC press (2018)
Kassab, M., DeFranco, Joanna F., Laplante, Phillip A.: Software testing: The state of the practice. *IEEE Softw.*, 34(5):46–52 (2017)
Le Cessie, S., Van Houwelingen, Johannes C.: Ridge estimators in logistic regression. *J. Royal Statist. Soc.: Series C (Applied Statistics)*, 41(1):191–201 (1992)
Malhotra, R.: A systematic review of machine learning techniques for software fault prediction. Appl. Soft Comput. **27**, 504–518 (2015)
Osman, H., Ghafari, M., Nierstrasz, O.: Automatic feature selection by regularization to improve bug prediction accuracy. In *2017 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)*, pages 27–32. IEEE (2017)
Pendharkar, Parag C.: A data envelopment analysis-based approach for data preprocessing. *IEEE Trans. Knowl. Data Eng.*, 17(10):1379–1388 (2005)
Rathore, Santosh ., Kumar, S.: Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems. Knowledge-Based Syst **119**, 232–256 (2017)
Shirabad, J.S., Menzies, T.: The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada (2005)
Sommerville, I.: *Software engineering*, 10th edn. Pearson Education (2016)
Son, Le H., Pritam, N., Khari, M., Kumar, R., Phuong, Pham Thi M., Thong, Pham H., et al.: Empirical study of software defect prediction: A systematic mapping. *Symmetry*, 11(2):212 (2019)
Strategic Planning. The economic impacts of inadequate infrastructure for software testing. *Nat. Inst. Standards Technol.* (2002)
Sunil, Jinu M., Kumar, L., Neti, Lalita Bhanu M.: Bayesian logistic regression for software defect prediction (s). In *SEKE*, pages 421–420 (2018)
Tassey, G.: The economic impacts of inadequate infrastructure for software testing. national institute of standards and technology, 2002. *Forschungsbericht (Zitiert auf Seite 2)* (1996)
Taylor, P.: *Autonomic Business Processes*. PhD thesis, University of York (2015)

Authors and Affiliations

Aftab Ali[1] · Naveed Khan[1] · Mamun Abu-Tair[1] · Joost Noppen[2] · Sally McClean[1] · Ian McChesney[1]

Naveed Khan
n.khan@ulster.ac.uk

Mamun Abu-Tair
m.abu-tair@ulster.ac.uk

Joost Noppen
johannes.noppen@bt.com

Sally McClean
si.mcclean@ulster.ac.uk

Ian McChesney
ir.mcchesney@ulster.ac.uk

[1] School of Computing, Ulster University, BT37 0QB Newtownabbey, UK

[2] Applied Research, BT, Ipswich, UK