The Seybold Report

ISSN 1533-9211

DOI 10.5281/zenodo.7313541

NEW SOFTWARE DEFECT PREDICTION METHOD BASED ON PCA AND OPTIMIZED LSTM

YAHYA KHALID ALI[1] and ZAID HAMODAT[2]

[1,2]Electrical and Computer Engineering, Altinbas University, Istanbul, Turkey.
Email: 203720051@org.altinbas.edu.tr[1], zaid.hamodat@altinbas.edu.tr[2]

Abstract

It is natural for the developed software to contain some defects and errors. The important issue is to detect these errors. Tests play an important role in detecting errors, but this is not always sufficient. Therefore, effective methods are needed to detect software errors and defects. The detecting error before publishing and serving the software the is main aim of the companies and the software engineers but this issue remain difficult and challenging issue. In this study, new method based KNN, and parameter optimization techniques presented in the field of software defect prediction. The proposed method applied ensemble learning with KNN based Grid search to present new model for software defect detection. Furthermore, several classifiers are applied to obtain best detection rate. The obtained results are compared and discussed to select best model. The presented method obtained suitable results when compared with traditional techniques. The three datasets that are used in this study are CM1, JM1, and PC1 which proposed method presented various results with these datasets vary between 75% to 94%.

1. INTRODUCTION

Today, technology is advancing rapidly and the software world is also developing with these advances. Despite these developments, the software revealed by the developers may contain some errors. Errors in software can generally be grouped under three categories; design errors; are typically errors in the design of the software architecture, implementation and delivery errors; these are errors that occur after the software developer's delivery to the customer. Detection of these errors at an early stage plays a major role. Software bugs; It causes developers to review the faulty modules in the projects repeatedly and as a result spend more time on the developed modules. Early detection of bugs in software; It ensures that the software is error-free and of higher quality before it is delivered to the customer [1].

When a software defect is encountered at any stage of the software development lifecycle, if the software development lifecycle is Waterfall, the analysis returns to the first step. A worse situation is that this situation can be encountered after delivery to the customer. In this case, which is encountered after delivery to the customer, the same step is repeated. Therefore, software maintenance costs increase significantly. In addition, trust problems may arise for customers who encounter software errors. It is important to constantly test and verify the source code in ensuring the quality of the developed software. According to Boehm, the cost calculated after the delivery of a software that is in a condition to be delivered to the end user is higher than the cost calculated during the development of the software. Software testing tools are used to detect errors in developed software. With test tools, the accuracy of the software is ensured before it is delivered to the customer. Test tools may not always be sufficient to catch the errors that occur [2].

The Seybold Report
ISSN 1533-9211
DOI 10.5281/zenodo.7313541

As the dependency on software increases, software quality becomes more important today. Software; It has been used almost everywhere and at every stage of life. Results that affect the software, such as errors, can lead to customer dissatisfaction and reduce the quality of the software. A software error causes both loss of time and financial losses [3]. The experience of a software developer when he/she first develops is not the same as the ability to predict errors in the software he/she develops after he/she develops himself/herself and gains experience. In order to achieve this, it is necessary to know which programs are more error-prone than others, and at which stages the probability of error is higher. Previous studies have shown that 27% man-hours were consumed by testing during the overall development process [4, 5].

2. LITERATURE REVIEW

Pelayo et al. [6] The Smote algorithm was applied simultaneously to different percentages of the dataset used to rank software bugs, increase minority class, and decrease majority class. They measured the success of the classification of the G-means of the C4.5 decision tree algorithm. This showed that the probability of success of the classification was increased. In this study, multiclass data was extracted from the dataset. Instead of shrinking the existing majority, only a few were enlarged. The software metrics of the new dataset obtained in this way were examined with regard to the information collection and its influence on algorithms such as C4.5 and PART [7].

Menzies et al. [8] used the quartile chart for software error estimation. Using data mining methods OneR, C4.5 and Naive Bayes, they developed an error estimation method with non-dynamic code features.

Lessmann et al. [9] compared classification success for software error prediction using 22 classifiers and 10 publicly available software error prediction datasets created by NASA. In the results obtained, they stated that the metric-based classifiers gave successful results, but there were no substantial changes in difference in conditions of the success story of the classifiers. In this study, data preprocessing was used to the datasets with the Thumped procedure.

Japkowicz et al. [10] deal with randomness, data increase, data reduction and cost control suggested above to solve the class imbalance problem. They studied the diversity of classification success with the size of the training sample and the level of class imbalance. The general results obtained showed that the distributions of unbalanced classes influence the results of decision trees, other neural networks and supporting vector machines.

Batista et al. [11] performed experiments measurement learning working on 13 datasets by improving artificial data and reducing artificial data applying the techniques they named Smote + Tomek and Smote + ENN. In the general findings found, it has been shown that artificial data augmentation techniques are advanced to artificial data saving techniques.

He et al. [12] developed ADASYN: adaptive synthetic data sampling approach for the unbalanced learning problem. Different minority a weighted distribution was preferred for the difficulties experienced in the learning process according to the class percentage levels. Thus,

The Seybold Report
ISSN 1533-9211
DOI 10.5281/zenodo.7313541

the tendency resulting from class imbalance has been reduced and it has been shown that the limits of the classifier have been increased to include difficult examples.

3. PROPOSED METHOD

In this study, new method based KNN, and parameter optimization techniques presented in the field of software defect prediction. The proposed method applied ensemble learning with KNN based Gridsearch to present new model for software defect detection. The proposed method consists from several steps to analyses the datasets by training and testing stages.

i. Start the simulation that written using Jupyter notebook. The proposed method executed using Jupyter notebook with python language. The aim of using these tools because of these tools are easy to use and installation.

ii. Read datasets that are used in this study, for example in this research three datasets used CM1, JM1, and PC1. These datasets are contain the features that can train the classifier to detect bugs (defects) in the software projects.

iii. The dataset divided into train and test which is main steps to develop high quality performance models.

iv. Train the KNN by using the training section of the dataset. And the Grid Search optimize the structure of the KNN to present best performance.

v. After the best structure parameters selected by the Grid Search than the test stage started to evaluate and test the performance of the model. This section mathematically represented as shown in equation 3.1.

$$\text{Euclidean } \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \quad (1)$$

$$\text{Manhattan } \sum_{i=1}^{k} |x_i - y_i| \quad (2)$$

$$\text{Minkowski } \left(\sum_{i=1}^{k} (|x_i - y_i|)^q\right)^{\frac{1}{q}} \quad (3)$$

These, equation explained the calculating the KNN classifier by calculating the distance between the tests point and the example. The y_i represented each feature point in the example dataset. The x_i represented the feature point in the test data. After the finding nearest points to the tested case the K number selected and the class type of the example estimated according to the nearest examples.

vi. In training and testing sections four classifiers in addition to our method is also trained and tested to find the best model.

vii. The accuracy of the models calculated.

viii. The execution completes after the results presented.

ISSN 1533-9211

DOI 10.5281/zenodo.7313541

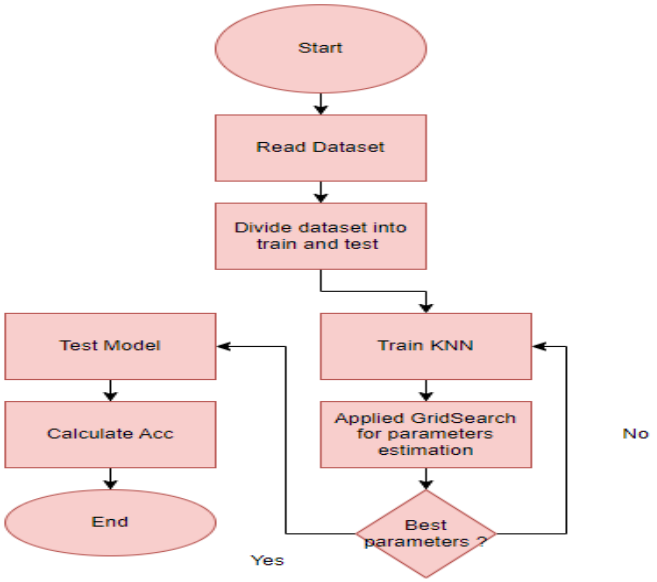

Figure 1: Proposed Method

4. RESULTS

Datasets are the name given to the structures in which raw data about a subject is stored. The data sets that everyone can easily access are increasing day by day. This includes datasets created for software error estimation. One of them is the publicly available PROMISE dataset owned by space explorer NASA. In this study, CM1, JM1, and PC1 datasets from the PROMISE dataset were used [40].

The results of all classifiers based the CM1 dataset presented in the figure 2.

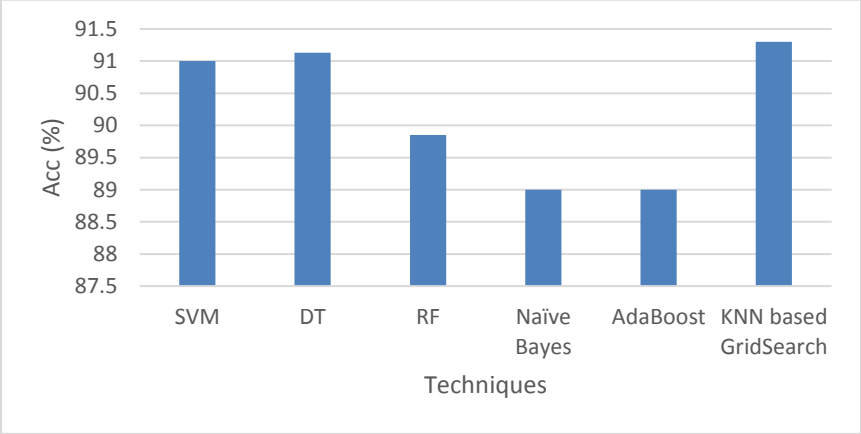

Figure 2: Classifier results with CM1

ISSN 1533-9211

DOI 10.5281/zenodo.7313541

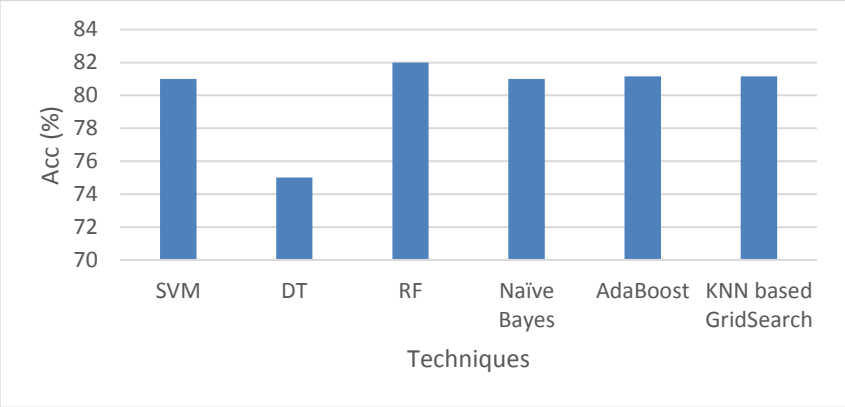

Figure 3: Classifier results with JM1

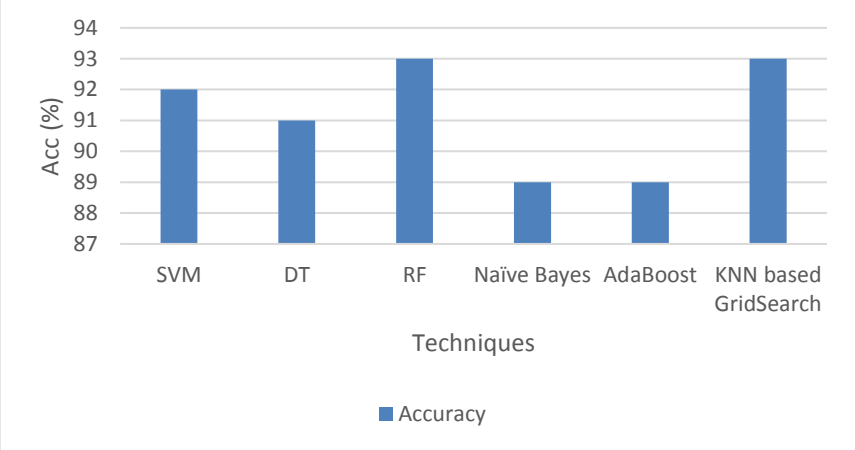

Figure 4: Classifier results with PC1

When analyzing the three datasets show that the proposed method presented suitable results when compared with previous studies. In CM1 dataset the proposed method presented best results than other with more than 90%. In JM1 dataset the all datasets presented low classification rate according to the first dataset. The accuracy rate of the dataset with all classifiers and our method vary between 70%-81% accuracies. The proposed method presented little bit low results from random forest. On the other hand, in the last dataset the proposed method also presented highest results with the random forest. The KNN is effective with low number of dataset and can learn easily and fast. The GridSearch technique is assist the classifier to enhance its classification rate compared with classic KNN.

5. CONCLUSIONS

Software defect prediction is one of the most important processes in software engineering. Accurate evaluation of bugs in software (especially its modules) is critical to creating bug-free software or software with as few bugs as possible. By using appropriate predictive models, software managers can improve the productivity of their employees. It can spend more time

The Seybold Report
ISSN 1533-9211
DOI 10.5281/zenodo.7313541

not only on testing, but also on maintaining software systems. Thus, the costs will be at a higher level.

In this study, several classifiers applied to detect software defects and the results compared to select best classifier. In the second part of the applications in the thesis study, the performance of 6 machine learning algorithms in Python language in Jupiter environment on NASA datasets and experimental datasets was evaluated. In this study the K nearest Neighbour algorıthm (KNN) method is a non-parametric classification and regression algorithm. Their ease of use and simple mathematical rationale generalize the use of predictive models in many other areas. It is based on the idea that the outcome of a case is the same as the outcome of the next case. Using a training set based on past observations, determine the dependent variable resulting from each item (observation) in the data set. The estimate of the future prediction is the average of the results of the current case compared to the results of subsequent elements in the training sample. Generally, the nearest observation is defined as the smallest Euclidean distance from the data point in question. The parameters of the KNN estimated using Grid Search to obtain best results. The obtained results vary between 81%-94% which is best of other classifiers.

In the future works, the researcher can used real datasets to evaluate the proposed method or the researcher can present new datasets as new academic studies. Furthermore, ensemble deep learning techniques also can apply as new study. The using several deep learning techniques such as deep belief network, deep sparse auto encoders, deep convolutional neural network, and deep recurrent network, long short-term memory, all these techniques can applied in one study to present new robust model that present best classification rate. Furthermore, feature selection techniques such as k mean and PCA applied to enhance the performance of the classifier to obtain best model that obtain maximum accuracy with low processing time.

References

1. V. Mitra, H. Franco, M. Graciarena and D. Vergyri, "Medium-duration modulation cepstral feature for robust speech recognition," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, 2014, pp. 1749-1753, doi: 10.1109/ICASSP.2014.6853898.
2. O. Hazrati, S. Ghaffarzadegan and J. H. L. Hansen, "Leveraging automatic speech recognition in cochlear implants for improved speech intelligibility under reverberation," 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, QLD, 2015, pp. 5093-5097, doi: 10.1109/ICASSP.2015.7178941.
3. R. Lotfidereshgi and P. Gournay, "Biologically inspired speech emotion recognition," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 5135-5139, doi: 10.1109/ICASSP.2017.7953135.
4. Y. Suh et al., "Development of distant multi-channel speech and noise databases for speech recognition by in-door conversational robots," 2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA), Seoul, 2017, pp. 1-4, doi: 10.1109/ICSDA.2017.8384419.
5. P. S. Nidadavolu, V. Iglesias, J. Villalba and N. Dehak, "Investigation on Neural Bandwidth Extension of Telephone Speech for Improved Speaker Recognition," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, 2019, pp. 6111-6115, doi: 10.1109/ICASSP.2019.8682992.

The Seybold Report
ISSN 1533-9211
DOI 10.5281/zenodo.7313541

6. Pelayo L, Dick S. “Applying novel resampling strategies to software defect prediction”. Fuzzy Information Processing Society NAFIPS ’07, San Diego, USA, 24-27 June, 2007.
7. Frank E, Witten IH. “Generating accurate rule sets without global optimization”. 15th International Conference on Machine Learning, Wisconsin, USA, 24-27 July 1998.
8. Menzies T, Greenwald J, Frank A. “Data mining static code attributes to learn defect predictors”. IEEE Transactions on Software Engineering, 33(1), 2-13, 2007.
9. Lessmann S, Baesens B, Mues C, Pietsch S. “Benchmarking classification models for software defect prediction: A proposed framework and novel findings”. IEEE Transactions on Software Engineering, 34(4), 485-496, 2008.
10. Japkowicz N, Stephen S. “The class imbalance problem: A systematic study”. Intelligent Data Analysis, 6(5), 429-449, 2002.
11. Batista G, Prati R, Monard M, “A Study of the Behavior of several methods for balancing machine learning training data”. ACM SIGKDD Explorations Special issue on learning from imbalanced datasets, 6(1), 20-29, 2004.
12. He H, Bai Y, Garcia EA, Li S. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. 2008 IEEE International Joint Conference on Neural Networks, Hong Kong, China, 1-8 June 2008.