JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452

Software Defects Classification Using RNN Model

Sweety Kataria[1],Prof. V. V. Subrahmanyam[2]
[1]Associate Professor, Kalindi College.
[2]School of Computer & Information Sciences (SOCIS), IGNOU.

Received 2022 April 02**; Revised** 2022 May 20**; Accepted** 2022 June 18**.**

Abstract
For development of software, controlling the quality is a prime concern of developers. Machine learning (ML) techniques enablesoftware engineers tocarry research in defect prediction that relies primarily on hand-crafted attributes, which are used to classify defect code in ML classifications. In the present studyML approach, a Recurrent Neural Network (RNN) for predicting software defect were used on PROMISE dataset for five different version of software defects.The proposed approach is compared with two existing approaches i.e. RF & SVM to predict software faults on historical data. Experiments on various Pledge datasets from the PROMISE repository: JM1, KC1, KC2, PCI and CM1 variants are studied. The proposed approach is better than the two existing support vector machine and random forest approaches in analysis. The RNN model in the present study shows accuracy in the range of 93.74 to 95.9 %. It shows maximum accuracy in PC1, and least in JM1. Further, the proposed approach gives precision in the range of 91.39 to 95.28 %. It shows highest precision in CM1, and least in PC1. Similarly, Recall is observed in the range of 92.21 to 94.64 % and thehighest recall is observed in JM1, and least in KC2.
Keywords: PROMISE Dataset, RNN, Machine Learning, Deep Learning.

INTRODUCTION

The software defects, imperfections during software development would make the desired software to fail to meet the actual requirements or may cause an unexpected result. Software defectsare error or mistakes, or may be bugin a software program that may cause unexpected outcome, or results malfunctioning in software. This error in the program may produce unintended outcome and prevent working as intended. Software defect prediction is localization oddeffects or modules have error in software. Software defects may significantly affect the software and may increase the software development cost due to expensesin identifying and correcting bugs. Defect prevention and identification are one of the critical part or important in software quality assurance. The quality of software depends on the possibilities of arising defects. High-risk elements should be predicting at the earliest to increase the efficiency of the software. The software defect detection and defect repair are the most time-consuming and costly in software development. Technically it is almost not feasible to remove each defect, but it is possible to reduce the defects in order to develop software modules with reliability and accuracy. The software defect predictionassistssoftware developers in detection of defects or

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452
software bugs.

In order to create prediction models, defect and measurement data must be obtained from real software development efforts to use as learning data. Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) have potential to beat state-of-the-art Deep Neural Network (DNN) systems[1], [2]. LSTMs are recurrent neural network have units known as memory blocks in the layers of recurrent hidden layer and are easier to train than standard RNNs[3], [4]. The Model must be measured by comparing the expected deficiency of the modules in the test against their real deficiency.Lessmann et al, [5]studied Benchmarking Classification Models for Software Defect Prediction. Recently some workers, [6] studied Software defect prediction using machine learning techniques and a systematic research analysis is conducted in which parameters of confusion, precision, recall, recognition accuracy, etc., are measured as well as compared with the prevailing schemes. ML based models are an important strategy for programming imperfection forecasts, these can be used for either classification or to estimate defect count/densities.Recent work demonstratedlearning by experience improvise these models on their predictive accuracy by adjusting their value of parameters[7]. The analytical analysis indicates that the proposed approach will provide more useful solutions for device defects prediction.

The software defect prediction to construct a prediction model, defect and data are required from development for learning process, the step labelling requiresdata for the training of the prediction model. The Prediction Models, Support Vector Machines or Bayesian Network or other General machine learners are used for a Prediction Model construction using a training set. In the Assessment step, a prediction model is evaluated using test data in addition to a training set by comparing the prediction and real labels[8]. To develop modules with reliability and accuracy, software defect prediction is required to assist developers in finding potential bugs and allocating their testing efforts. In order to create these prediction models, defect and measurement data must be obtained from real software development efforts to use it as a learning collection.

MOTIVATION

Software Defect prediction is the critical requirement of software quality and reliability. For detecting error prone modules in an intensive system and the Software Defect prediction serves as a dynamic and self-motivated research domain in software development for efficient management of the software quality. The model can be developed by prior resources and can predict defect prone modules.

RELATED WORK

Akiyama[9] built the first defect prediction model to predict the number of defects based on lines of code and built a very simple model using LOC. Tripathiand Rai in,[10]presenteda comparative analysis between traditional and Machine Learning methods and show that ML approaches have a more reliable estimate of effort relative to conventional methods of

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452

estimating effort. Bansal et al.[11]reported that the production of a fuzzy multi-criteria-based approach to decision-making by combining Fuzzy Set Theory. Muketha[12] exhaustively study current software commitment calculation approaches by developing calculation methods tailored to modernise app creation techniques.Lessmann et al, [5]studied Benchmarking Classification Models for Software Defect Prediction. Prabha and Shivakumar[6] studied Software defect prediction using machine learning techniques and A systematic research analysis is conducted in which parameters of confusion, precision, recall, recognition accuracy, etc.are measured as well as compared with the prevailing schemes.

Przemyslaw Pospieszny *et al.* [13]*in 2018 reported* Software effort and duration estimation using ISBSG dataset, intelligent data preparation, an average group of three machine learning algorithms and cross-validation ISBSG data collection. Research gap: The commitment and length calculation models obtained was supposed to provide a decision making method for designing organizations or integrating information systems as research gaps.

Dhingra*and Mann in 2014* [14]*worked on*neuro fuzzy model for estimating software time using Membership function models (Gaussian MF, Triangular MF and Trapezoidal MF). Research gap: Neuro Fuzzy model of Trapezoidal membership feature performs more than every other configuration.

Federica Sarro et al. in 2016 [15]*presented* Bi-objective effort estimation algorithm using Real-world datasets from the PROMISE repository, involving 724 different software projects in total. Research gap: The planned algorithm exceeds the standard, modernized well as all the three substitute formulations, which are statistically relevant ($p <; 0.001$) and with large effect size ($A_{12} \geq 0.9$) over all five datasets

Ochodek et al. in 2015 [16]*studied* using Function Point Analysis (FPA), Software Non-functional Assessment Process (SNAP) Measure the non-functional size of applications. Research gap: The study findings indicate that SNAP will help to alleviate certain well- FPA process limitations.

Ingold et al. in 2013[17] *work on* Constructive Rapid Application Development Model (CORADMO) Research gap:CORADMO attempts to measure the influence of primary program factors and thereby helps managers to predict the relative timetable arising from the variance of certain parameters.

Humayun and Gang in 2012[18] *work on* effort estimation using Global software development (GSD), artificial intelligence, machine learning. A qualitative study is performed between conventional approaches for calculating commitment and ML approaches. Research gap: Results indicate that ML methods provide us a more reliable measurement of effort relative to conventional methods of estimating effort.

Singhand Leavline *in 2019* [19]*studied* Dimensionality reduction using Feature subset selection and feature-ranking methods. Research gap: The techniques employed increase the classifier's prediction precision, reduce the incorrect prediction factor, as well as minimize the cost of time and space to construct the statistical model.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452

Shepperd *et al.*in 2012 [20]reported Evaluating prediction systems Using An unbiased statistic, Standardized Accuracy, Random 'predictions'. Research gap: Recently reported scientific evaluations of prediction processes are being re-examined and the initial findings found dangerous.

Malathiand Sridhar in *2012* [21]*work on* Software Cost Estimation using Quantitative basis for the development and validation. Research Gap: By constant analysis of different measures and methods, the findings are expected to change.

Brown and Boehm in 2010[22] *work on* Software Cost Estimation, using Directed System of Systems (DSOS) or Acknowledged Systems of Systems (ASOS).Research gap: The technique is being used to measure the expense of software production of applications.

The RNNs are helpful and popular tool for modeling complex sequence Data and have been utilized in various applications such as speech recognition, sentiment analysis, music, and video [[23]; [24]; [25]; [26]). Machine Learning based models are one of the strategies for programming imperfection forecasts. Similar to regression models, these can be used for either classification (defective/not defective) or to estimate defect count/densities. Over time as more data is made available, the models improvise on their predictive accuracy by adjusting their value of parameters (learning by experience)[7].

In the standard deep learning models of AI, convolutional neural networks (CNNs) are widely used for visual tasks such as classification and recognition task for images and for objects, while recurrent neural network (RNNs) are typically for tasks mainly involve temporal patterns fed into the network as sequential input. Many RNN models are tested to make learning methods easier and better. The LSTM (Long Short-Term Memory) model was designed specifically as an RNN for sequential machine learning tasks like speech recognition, language understanding and sequence to sequence translation[27]. The IndRNN, a new type of RNNs with the recurrent connection formulated as Hadamard product, referred to as independently recurrent neural network (IndRNN), where neurons in the same layer are independent of each other and connected across layers. As connections in CNNs to facilitate the gradient flow in the network and achieve state-of-the-art RNN performance [28]. The convolutional neural networks have applied to sequential tasks but have substantial network-size and data-history memory requirements[29], [30]

PROBLEM STATEMENT

Software defects affectthe quality of the software. A defective software module creates a massive impact on the quality and efficiency of the software with increase in cost also, increasesthe software development cost and time and results in development to a smaller extent. Defect prediction can be a valuable tool for guiding the use of tools for quality assurance for software. Many research projects covered methods for predicting defects and methodological aspects of prediction research, the real cost - saving potential of predicting defects remains unclear.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452

PROPOSED WORK

3.1 PROPOSED MODEL

To determine the accurate sized figures for defect in the software.

3.1.1LSTM-RNN ALGORITHM

Input: PROMISE Software Defect Prediction Dataset (features with Labels)
Output: Classification model

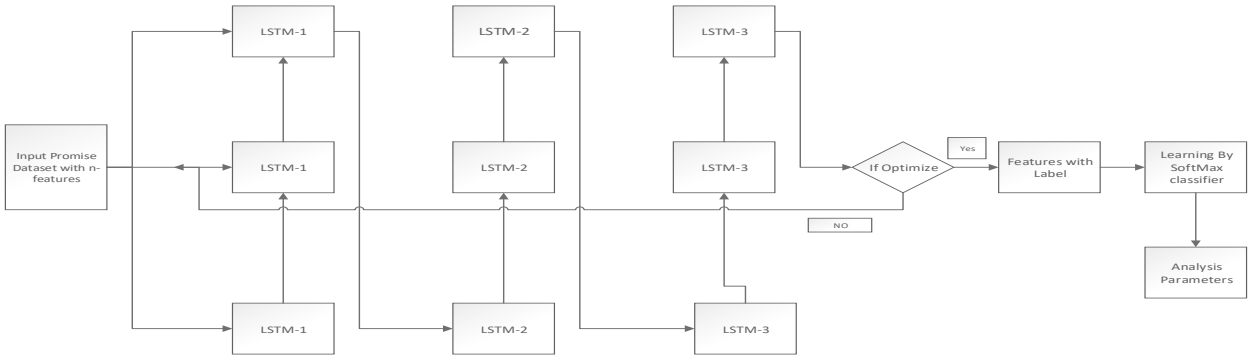
1. Start
2. Software defect prediction data store in NumPy 3 dimensional array (N, W,F)
   - N number of training Sequence
   - W is sequence Length
   - F is the number
3. A network structure is built with [1, a, b,1] dimension, where there is 1input layer, a neuron in the next layer, b neuron in the subsequent layer and a single layer with a linear activation function.
4. Train constructed LSTM-RNN network on the data
5. Use the output of last layer as the prediction of the statement.
6. Repeat 4 and 5 steps until optimize convergence.
7. Obtain prediction of testing data on trained LSTM-RNN network
8. Evaluate accuracy, precision and recall
9. End

3.2 PROPOSED FLOWCHART

In the present study ML approach, a Recurrent Neural Network (RNN) for predicting software defect were used on PROMISE dataset for five different version of software defects. The proposed approach is compared with two existing approaches i.e. RF & SVM to predict software faults on historical data. Experiments on various Pledge datasets from the PROMISE repository: JM1, KC1, KC2, PCI and CM1 variants are studied.

In the present study ML approach, a RNNfor predicting software defect were used on PROMISE dataset for five different version of software defect. The proposed flow chart include step 1, in which a network structure is built a input layer, a neuron in the next layer of the proposed model in flow chart and neuron in the subsequent layers and a single layer with activation function. The flow chart (Fig 3.1)showsconstructed LSTM-RNN network on the data and use the output as the last layer in the prediction of the statement. The steps 4 and 5 repeat until optimize convergence. Prediction of testing data obtained on trained LSTM-RNN network. The evaluation includes evaluate accuracy, precision and recall.

Fig 3.1 Proposed LSTM-RNN Approach



JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452

4. EXPERIMENT AND RESULTS

4.1 EXPERIMENTAL SETUP

The experiment setup details with different parameters like dataset, type of dataset models uses and metrics for analysis performance model are presented in table 4.1. The table shows experiment setup of the present study and represents details of used dataset and type of dataset used in the study. The type of dataset used are different version of software's i.e. CM1,JC1,KC1,KC2,PC1. In setup also show the algorithms and its component LSTM-RNN. LSTMs are recurrent neural network have memory blocks in the layers of recurrent hidden layer and are easier to train than standard RNNs [3], [4].

Table 4.1: Experimental Setup of Software Defect Prediction

Parameters	**Value**
Dataset	**Promise**
Activation function	**RELU**
Pooling	**Max**
Features	**20**
type of dataset	**CM1,JC1,KC1,KC2,PC1**
LSTM-RNN Layer	**3**
Classifier	**SoftMax**
Metrics	**Accuracy, precision, recall**

4.2RESULTS

The observations in the present study of performance analysis of LSTM approach and two existing ML approaches on different metrics and datasets are shown in table 4.2. The metrics used for the analysis are precision, recall and accuracy. As shown in the table 4.2, the proposed LSTM approach is better than the two existing support vector machine and random forest approaches in analysis. The dataset used for the comparison study are JM1, KC1, KC2, PC1 and CM1. The performance of LSTM approach on accuracy on different PROMISE datasets are presented using the bar diagram for comparison in between the different data sets and presented in the fig 1.The performance of LSTM approach on Precision metrics using different PROMISE dataset are presented by bar diagram in fig. 2. The table 4.3 shows analysis of proposed model on Recall using different PROMISE dataset through LSTM -RNN approach. The comparative performance analysis results of LSTM approach on different metrics and datasets are shown using bar diagram in fig 4. For metrics used precision, recall and accuracy,dataset using JM1, KC1, KC2, PC1 and CM1.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452

Table 4.2: Metrics Comparison of LSTM approach with SVM & RF, Dataset use JM1, KC1, KC2, PC1 and CM1

Dataset	Accuracy			Precision			Recall		
	(SVM)	(RF)	(LSTM)	(SVM)	(RF)	(LSTM)	(SVM)	(RF)	(LSTM)
CM1	72.22692308	83.04685898	94.04333333	72.76794872	83.39115384	95.28333333	72.66121795	83.41176282	94.27916667
JM1	72.17051282	83.08535257	93.74666667	71.6	82.09785256	93.91333333	72.06666667	82.64349359	94.64
KC1	72.5275641	83.59932692	93.89666667	70.63141026	81.19759616	92.24666667	71.13397436	81.88259615	92.73333333
KC2	73.29807692	84.41057692	94.675	70.51730769	81.14826923	91.395	71.44807692	82.29365385	92.215
PC1	73.76923077	84.83461538	95.9	70.73076923	81.34038462	91.95	71.96153846	82.75576923	93.55

Fig 4.1 Accuracy Analysis Comparison of Software defect prediction in LSTM on different PROMISE dataset

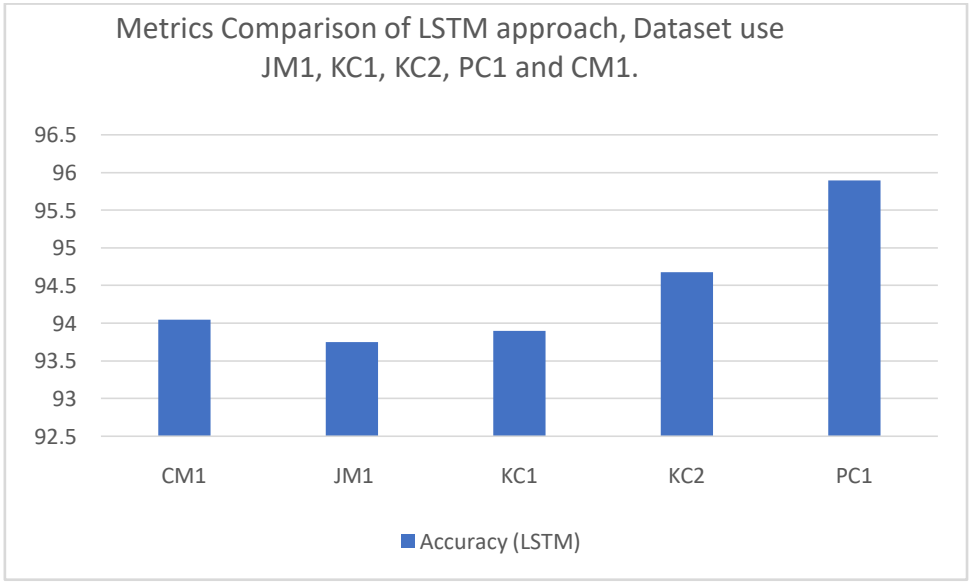


Fig 2: Precision Analysis Comparison of Software defect prediction in LSTM on different PROMISE dataset

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452

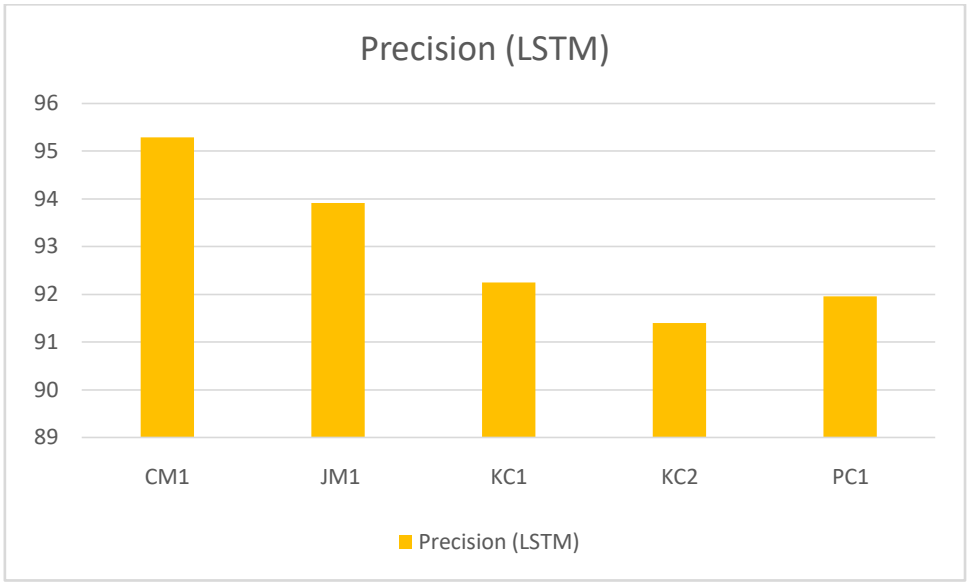


Fig 3 Recall Analysis Comparison of Software defect prediction in LSTM on different PROMISE dataset

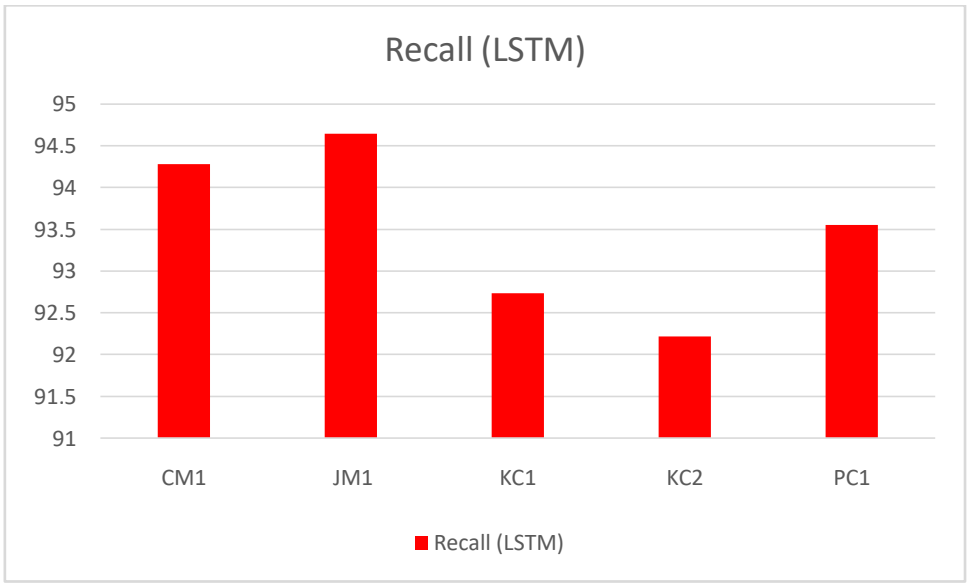
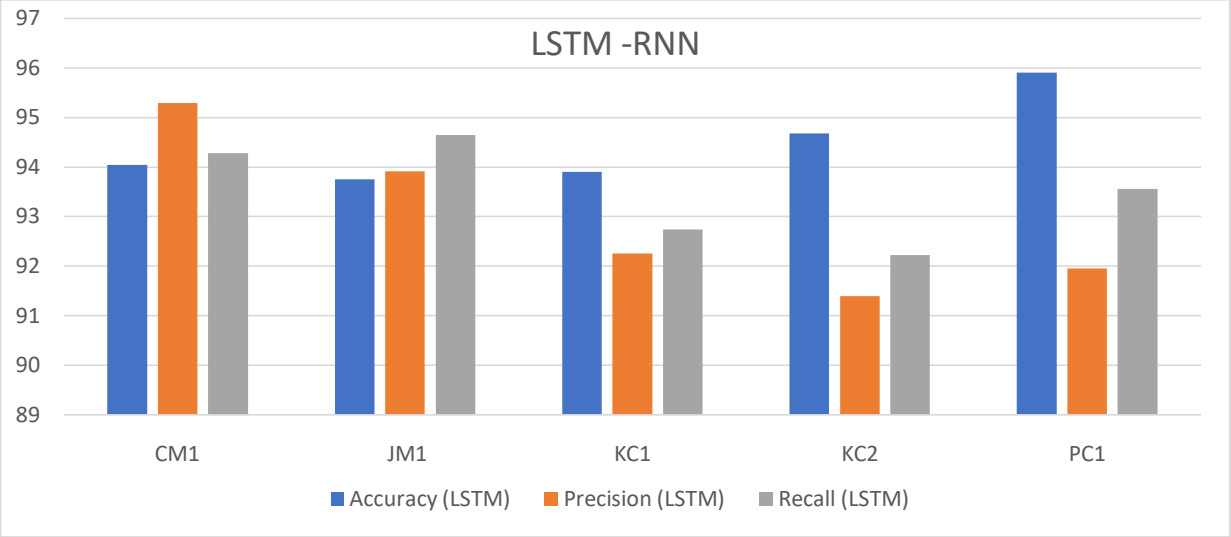


Fig 4: Metrics Comparison of LSTM approach, Dataset use JM1, KC1, KC2, PC1 and CM1.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452



4.3 RESULTS - ANALYSIS AND DISCUSSION

- In experiment use LSTM -RNN approachforpredicting software defect. In experiment, we used PROMISE Dataset using five different version of software defect.
- Experiment results use three performance metrics Precision, recall and Accuracy. These performance parameters representthe model reliability and validate model by experiment on different dataset.
- In model use a deep learning model using LSTM-RNN approach. This approach like CNN process not find any sequence by LSTM-RNN make sequence model, but the prediction model constructed can provide a satisfactory performance.The proposed approach is better than the two existing support vector machine and random forest approaches in analysis.
- In fig 4 shows the precision, recall and accuracy performance of models using LSTM-RNN. The Precision was found maximum in PC1 and shows perform efficiently in all datasets. The metrics recall performs most efficiently in JM1. On the other hand, Precision was found maximum in CM1 and perform efficiently. LSTM show nonlinear mapping on RELU Layer. Accuracy range 93.74 to 95.9 % and shows maximum accuracy in PC1, and least was detected in JM1. Precision range 91.39 to 95.28 % and shows highest accuracy in CM1, and least was detected in PC1. Recall range 92.21 to 94.64 % and shows highest accuracy in JM1, and least was detected in KC2 in to LSTM-RNN.

CONCLUSION

The technique LSTM-RNN might prove appropriate for binary classification tasks that contain components of non-parametric applied statistics, neural networks and machine learning. In the present study, a specialized approachfor machine learning a recurrent neural network for predicting software defect were used on PROMISE Dataset for five different version of software defect. The prediction model constructed can provide a satisfactory performance. These metric set can be helpful for software engineers and developers.The comparison results showed that the approach has the best results over the others. Moreover, experimental results showed that using RNN approach provides

a better performance for software defect prediction than RF and SVM approaches model.In conclude, RNN remark Accuracy range 93.74 to 95.9 % and shows maximum accuracy in PC1, and least was detected in JM1. Precision range 91.39 to 95.28 % and shows highest accuracy in CM1, and least was detected in PC1. Recall range 92.21 to 94.64 % and shows highest accuracy in JM1, and least was detected in KC2in to LSTM-RNN. Software defect Prediction features dependent to each other, In future enhance this work depend on Bayesian network which combined with deep learning. In future also increase the validation step by using validation approaches and also shall focus on improving the feature selection approaches.

ACKNOWLEDGEMENTS

We are gratefully acknowledging Kalindi College and University of Delhi for financial support. We also thank IGNOU, New Delhi for providing the facilities.

REFERENCES

[1] A. Graves, N. Jaitly, and A. Mohamed, “Hybrid speech recognition with deep bidirectional LSTM,” 2013, pp. 273–278.

[2] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition,” *ArXiv Prepr. ArXiv14021128*, 2014.

[3] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.

[5] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, “Benchmarking classification models for software defect prediction: A proposed framework and novel findings,” *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, 2008.

[6] C. L. Prabha and N. Shivakumar, “Software defect prediction using machine learning techniques,” 2020, pp. 728–733.

[7] M. K. Thota, F. H. Shajin, and P. Rajesh, “Survey on software defect prediction techniques,” *Int. J. Appl. Sci. Eng.*, vol. 17, no. 4, pp. 331–344, 2020.

[8] M. S. Rawat and S. K. Dubey, “Software defect prediction models for quality improvement: a literature study,” *Int. J. Comput. Sci. Issues IJCSI*, vol. 9, no. 5, p. 288, 2012.

[9] F. Akiyama, “An Example of Software System Debugging.,” 1971, vol. 71, pp. 353–359.

[10] R. Tripathi and P. Rai, “Machine learning methods of effort estimation and its performance evaluation criteria,” *IJCSMC*, vol. 6, no. 1, pp. 61–67, 2017.

[11] A. Bansal, B. Kumar, and R. Garg, “Multi-criteria decision making approach for the selection of software effort estimation model,” *Manag. Sci. Lett.*, vol. 7, no. 6, pp. 285–296, 2017.

[12] S. W. Munialo and G. M. Muketha, “A review ofagile software effort estimation methods,” 2016.

[13] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, “An effective approach for software project effort and duration estimation with machine learning algorithms,” *J. Syst. Softw.*, vol. 137, pp. 184–196, 2018.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 1208-1218
https://publishoa.com
ISSN: 1309-3452
[14] S. Dhingra and P. S. Mann, "Design and implementation of neuro fuzzy model for software development time estimation," *Int. J. Comput. Appl.*, vol. 86, no. 5, 2014.
[15] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," 2016, pp. 619–630.
[16] M. Ochodek and B. Ozgok, "Functional and Non-functional Size Measurement with IFPUG FPA and SNAP—Case Study," in *Software Engineering in Intelligent Systems*, Springer, 2015, pp. 19–33.
[17] D. Ingold, B. Boehm, and S. Koolmanojwong, "A model for estimating agile project process and schedule acceleration," 2013, pp. 29–35.
[18] M. Humayun and C. Gang, "Estimating effort in global software development projects using machine learning techniques," *Int. J. Inf. Educ. Technol.*, vol. 2, no. 3, p. 208, 2012.
[19] D. A. A. G. Singh and E. J. Leavline, "Dimensionality Reduction for Classification and Clustering," *Int. J. Intell. Syst. Appl. IJISA*, vol. 11, no. 4, pp. 61–68, 2019.
[20] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, 2012.
[21] S. Malathi and S. Sridhar, "Analysis of size metrics and effort performance criterion in software cost estimation," *Indian J. Comput. Sci. Eng.*, vol. 3, no. 1, pp. 24–31, 2012.
[22] A. W. BROWN and B. BOEHM, "Software cost estimation in the incremental commitment model," 2010, vol. 4, no. 01, pp. 45–55.
[23] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A Comparison of Sequence-to-Sequence Models for Speech Recognition.," 2017, pp. 939–943.
[24] D. Dwibedi, P. Sermanet, and J. Tompson, "Temporal reasoning in videos using convolutional gated recurrent units," 2018, pp. 1111–1116.
[25] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," 2017, pp. 2392–2396.
[26] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," 2015, pp. 1422–1432.
[27] A. Graves, "Generating sequences with recurrent neural networks," *ArXiv Prepr. ArXiv13080850*, 2013.
[28] S. Li, W. Li, C. Cook, and Y. Gao, "Deep independently recurrent neural network (indrnn)," *ArXiv Prepr. ArXiv191006251*, 2019.
[29] A. van den Oord *et al.*, "Wavenet: A generative model for raw audio," *ArXiv Prepr. ArXiv160903499*, 2016.
[30] K. Benidis *et al.*, "Neural forecasting: Introduction and literature overview," *ArXiv Prepr. ArXiv200410240*, 2020.