Software Bug Prediction using Machine Learning Approach

Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah
Information Technology Department
Mutah University, Al Karak, Jordan

***Abstract*—Software Bug Prediction (SBP) is an important issue in software development and maintenance processes, which concerns with the overall of software successes. This is because predicting the software faults in earlier phase improves the software quality, reliability, efficiency and reduces the software cost. However, developing robust bug prediction model is a challenging task and many techniques have been proposed in the literature. This paper presents a software bug prediction model based on machine learning (ML) algorithms. Three supervised ML algorithms have been used to predict future software faults based on historical data. These classifiers are Naïve Bayes (NB), Decision Tree (DT) and Artificial Neural Networks (ANNs). The evaluation process showed that ML algorithms can be used effectively with high accuracy rate. Furthermore, a comparison measure is applied to compare the proposed prediction model with other approaches. The collected results showed that the ML approach has a better performance.**

Keywords—Software bug prediction; faults prediction; prediction model; machine learning; Naïve Bayes (NB); Decision Tree (DT); Artificial Neural Networks (ANNs)

I. INTRODUCTION

The existence of software bugs affects dramatically on software reliability, quality and maintenance cost. Achieving bug-free software also is hard work, even the software applied carefully because most time there is hidden bugs. In addition to, developing software bug prediction model which could predict the faulty modules in the early phase is a real challenge in software engineering.

Software bug prediction is an essential activity in software development. This is because predicting the buggy modules prior to software deployment achieves the user satisfaction, improves the overall software performance. Moreover, predicting the software bug early improves software adaptation to different environments and increases the resource utilization.

Various techniques have been proposed to tackle Software Bug Prediction (SBP) problem. The most known techniques are Machine Learning (ML) techniques. The ML techniques are used extensively in SBP to predict the buggy modules based on historical fault data, essential metrics and different software computing techniques.

In this paper, three supervised ML learning classifiers are used to evaluate the ML capabilities in SBP. The study discussed Naïve Bayes (NB) classifier, Decision Tree (DT) classifier and Artificial Neural Networks (ANNs) classifier. The discussed ML classifiers are applied to three different datasets obtained from [1] and [2] works.

In addition to, the paper compares between NB classifier, DT classifier and ANNs classifier. The comparison based on different evaluation measures such as accuracy, precision, recall, F-measures and the ROC curves of the classifiers.

The rest of this paper is organized as follow. Section 2 presents a discussion of the related work in SBP. An overview of the selected ML algorithms is presented in Section 3. Section 4 describes the datasets and the evaluation methodology. Experimental results are shown in Section 5 followed by conclusions and future works.

II. RELATED WORK

There are many studies about software bug prediction using machine learning techniques. For example, the study in [2] proposed a linear Auto-Regression (AR) approach to predict the faulty modules. The study predicts the software future faults depending on the historical data of the software accumulated faults. The study also evaluated and compared the AR model and with the Known power model (POWM) used Root Mean Square Error (RMSE) measure. In addition to, the study used three datasets for evaluation and the results were promising.

The studies in [3], [4] analyzed the applicability of various ML methods for fault prediction. Sharma and Chandra [3] added to their study the most important previous researches about each ML techniques and the current trends in software bug prediction using machine learning. This study can be used as ground or step to prepare for future work in software bug prediction.

R. Malhotra in [5] presented a good systematic review for software bug prediction techniques, which using Machine Learning (ML). The paper included a review of all the studies between the period of 1991 and 2013, analyzed the ML techniques for software bug prediction models, and assessed their performance, compared between ML and statistic techniques, compared between different ML techniques and summarized the strength and the weakness of the ML techniques.

In [6], the paper provided a benchmark to allow for common and useful comparison between different bug prediction approaches. The study presented a comprehensive comparison between a well-known bug prediction approaches, also introduced new approach and evaluated its performance by building a good comparison with other approaches using the presented benchmark.

D. L. Gupta and K. Saxena [7] developed a model for object-oriented Software Bug Prediction System (SBPS). The study combined similar types of defect datasets which are available at Promise Software Engineering Repository. The study evaluated the proposed model by using the performance measure (accuracy). Finally, the study results showed that the average proposed model accuracy is 76.27%.

Rosli et al. [8] presented an application using the genetic algorithm for fault proneness prediction. The application obtains its values, such as the object-oriented metrics and count metrics values from an open source software project. The genetic algorithm uses the application's values as inputs to generate rules which employed to categorize the software modules to defective and non-defective modules. Finally, visualize the outputs using genetic algorithm applet.

The study in [9] assessed various object-oriented metrics by used machine learning techniques (decision tree and neural networks) and statistical techniques (logical and linear regression). The results of the study showed that the Coupling Between Object (CBO) metric is the best metric to predict the bugs in the class and the Line Of Code (LOC) is fairly well, but the Depth of Inheritance Tree (DIT) and Number Of Children (NOC) are untrusted metrics.

Singh and Chug [10] discussed five popular ML algorithms used for software defect prediction i.e. Artificial Neural Networks (ANNs), Particle Swarm Optimization (PSO), Decision Tree (DT), Naïve Bayes (NB) and Linear Classifiers (LC). The study presented important results including that the ANN has lowest error rate followed by DT, but the linear classifier is better than other algorithms in term of defect prediction accuracy, the most popular methods used in software defect prediction are: DT, BL, ANN, SVM, RBL and EA, and the common metrics used in software defect prediction studies are: Line Of Code (LOC) metrics, object oriented metrics such as cohesion, coupling and inheritance, also other metrics called hybrid metrics which used both object oriented and procedural metrics, furthermore the results showed that most software defect prediction studied used NASA dataset and PROMISE dataset.

Moreover, the studies in [11], [12] discussed various ML techniques and provided the ML capabilities in software defect prediction. The studies assisted the developer to use useful software metrics and suitable data mining technique in order to enhance the software quality. The study in [12] determined the most effective metrics which are useful in defect prediction such as Response for class (ROC), Line of code (LOC) and Lack Of Coding Quality (LOCQ).

Bavisi et al. [13] presented the most popular data mining technique (k-Nearest Neighbors, Naïve Bayes, C-4.5 and Decision trees). The study analyzed and compared four algorithms and discussed the advantages and disadvantages of each algorithm. The results of the study showed that there were different factors affecting the accuracy of each technique; such as the nature of the problem, the used dataset and its performance matrix.

The researches in [14], [15] presented the relationship between object-oriented metrics and fault-proneness of a class. Singh et al. [14] showed that CBO, WMC, LOC, and RFC are effective in predicting defects, while Malhotra and Singh [15] showed that the AUC is effective metric and can be used to predict the faulty modules in early phases of software development and to improve the accuracy of ML techniques.

This paper discusses three well-known machine learning techniques DT, NB and ANNs. The paper also evaluates the ML classifiers using various performance measurements (i.e. accuracy, precision, recall, F-measure and ROC curve). Three public datasets are used to evaluate the three ML classifiers.

On the other hand, most of the mentioned related works discussed more ML techniques and different datasets. Some of the previous studies mainly focused on the metrics that make the SBP as efficient as possible, while other previous studies proposed different methods to predict software bugs instead of ML techniques.

III. USED MACHINE LEARNING ALGORITHMS

The study aims to analyze and assess three supervised Machine Learning algorithms, which are Naïve Bayes (NB), Artificial Neural Network (ANN) and Decision Tree (DT). The study shows the performance accuracy and capability of the ML algorithms in software bug prediction and provides a comparative analysis of the selected ML algorithms.

The supervised machine learning algorithms try to develop an inferring function by concluding relationships and dependencies between the known inputs and outputs of the labeled training data, such that we can predict the output values for new input data based on the derived inferring function. Following are summarized description of the selected supervised ML algorithms:

- *Naïve Bayes (NB)*: NB is an efficient and simple probabilistic classifier based on Bayes theorem with independence assumption between the features. NB is not single algorithms, but a family of algorithms based on common principle, which assumes that the presence or absence of a particular feature of the class is not related to the presence and absence of any other features [16], [17].

- *Artificial Neural Networks (ANNs)*: ANNs are networks inspired by biological neural networks. Neural networks are non-linear classifier which can model complex relationships between the inputs and the outputs. A neural network consists of a collection of processing units called neurons that are work together in parallel to produce output [16]. Each connection between neurons can transmit a signal to other neurons and each neuron calculates its output using the nonlinear function of the sum of all neuron’s inputs.

- *Decision Tree (DT)*: DT is a common learning method used in data mining. DT refers to a hierarchal and predictive model which uses the item’s observation as branches to reach the item’s target value in the leaf. DT is a tree with decision nodes, which have more than one branch and leaf nodes, which represent the decision.

TABLE II. DS1 - THE FIRST SOFTWARE FAULTS DATASET

D_i	F_i	T_i	D_i	F_i	T_i
1	2	75	24	2	8
2	0	31	25	1	15
3	30	63	26	7	31
4	13	128	27	0	1
5	13	122	28	22	57
6	3	27	29	2	27
7	17	136	30	5	35
8	2	49	31	12	26
9	2	26	32	14	36
10	20	102	33	5	28
11	13	53	34	2	22
12	3	26	35	0	4
13	3	78	36	7	8
14	4	48	37	3	5
15	4	75	38	0	27
1`6	0	14	39	0	6
17	0	4	40	0	6
18	0	14	41	0	4
19	0	22	42	5	0
20	0	5	43	2	6
21	0	9	44	3	5
22	30	33	45	0	8
23	15	118	46	0	2

IV. DATASETS AND EVALUATION METHODOLOGY

The used datasets in this study are three different datasets, namely DS1, DS2 and DS3. All datasets are consisting of two measures; the number of faults (F_i) and the number of test workers (T_i) for each day (D_i) in a part of software projects lifetime. The DS1 dataset has 46 measurements that involved in the testing process presented in [1]. DS2, also taken from [1], which measured a system faults during 109 successive days of testing the software system that consists of 200 modules with each having one kilo line of code of Fortran. DS2 has 111 measurements. DS3 is developed in [2], which contains real measured data for a test/debug program of a real-time control application presented in [18]. Tables I to III present DS1, DS2 and DS3, respectively.

The datasets were preprocessed by a proposed clustering technique. The proposed clustering technique marks the data with class labels. These labels are set to classify the number of faults into five different classes; A, B, C, D, and E. Table IV shows the value of each class and number of instances that belong to it in each dataset.

In order to evaluate the performance of using ML algorithms in software bug prediction, we used a set of well-known measures [19] based on the generated confusion matrixes. The following subsections describe the confusion matrix and the used evaluation measures.

TABLE III. DS2 - THE SECOND SOFTWARE FAULTS DATASET

D_i	F_i	T_i	D_i	F_i	T_i	D_i	F_i	T_i
1	5	4	38	15	8	75	0	4
2	5	4	39	7	8	76	0	4
3	5	4	40	15	8	77	1	4
4	5	4	41	21	8	78	2	2
5	6	4	42	8	8	79	0	2
6	8	5	43	6	8	80	1	2
7	2	5	44	20	8	81	0	2
8	7	5	45	10	8	82	0	2
9	4	5	46	3	8	83	0	2
10	2	5	47	3	8	84	0	2
11	31	5	48	8	4	85	0	2
12	4	5	49	5	4	86	0	2
13	24	5	50	1	4	87	2	2
14	49	5	51	2	4	88	0	2
15	14	5	52	2	4	89	0	2
16	12	5	53	2	4	90	0	2
17	8	5	54	7	4	91	0	2
18	9	5	55	2	4	92	0	2
19	4	5	56	0	4	93	0	2
20	7	5	57	2	4	94	0	2
21	6	5	58	3	4	95	0	2
22	9	5	59	2	4	96	1	2
23	4	5	60	7	4	97	0	2
24	4	5	61	3	4	98	0	2
25	2	5	62	0	4	99	0	2
26	4	5	63	1	4	100	1	2
27	3	5	64	0	4	101	0	1
28	9	6	65	1	4	102	0	1
29	2	6	66	0	4	103	1	1
30	5	6	67	0	4	104	2	1
31	4	6	68	1	3	105	0	1
32	1	6	69	1	3	106	1	2
33	4	6	70	0	3	107	0	2
34	3	6	71	0	3	108	0	1
35	6	6	72	1	3	109	1	1
36	13	6	73	1	4	110	0	1
37	19	8	74	0	4	111	1	1

A. *Confusion Matrix*

The confusion matrix is a specific table that is used to measure the performance of ML algorithms. Table V shows an example of a generic confusion matrix. Each row of the matrix represents the instances in an actual class, while each column represents the instance in a predicted class or vice versa. Confusion matrix summarizes the results of the testing algorithm and provides a report of the number of True Positive (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

TABLE IV. DS3 - THE THIRD SOFTWARE FAULTS DATASET

D_i	F_i	T_i	D_i	F_i	T_i	D_i	F_i	T_i
1	4	1	38	9	2	75	1	2
2	0	1	39	7	2	76	11	2
3	7	1	40	12	2	77	1	2
4	10	1	41	12	2	78	0	2
5	13	1	42	15	2	79	2	2
6	8	1	43	14	2	80	2	2
7	13	1	44	7	2	81	4	2
8	4	1	45	9	2	82	1	2
9	7	1	46	11	2	83	0	2
10	8	1	47	5	2	84	4	2
11	1	1	48	7	2	85	1	1
12	6	1	49	7	2	86	1	1
13	13	1	50	14	2	87	0	1
14	7	1	51	13	2	88	2	3
15	9	1	52	14	2	89	0	1
16	8	2	53	11	2	90	0	2
17	5	2	54	2	1	91	1	1
18	10	2	55	4	1	92	1	1
19	7	2	56	4	2	93	0	1
20	11	2	57	3	2	94	0	2
21	5	2	58	6	2	95	0	1
22	8	2	59	6	2	96	0	1
23	13	2	60	2	2	97	1	2
24	9	2	61	0	1	98	0	1
25	7	2	62	0	1	99	1	1
26	7	2	63	3	1	100	0	1
27	5	2	64	0	1	101	0	1
28	7	2	65	4	1	102	0	2
29	6	1	66	0	1	103	0	1
30	6	1	67	1	1	104	2	1
31	4	1	68	2	1	105	0	1
32	12	2	69	0	2	106	1	2
33	6	2	70	1	2	107	0	2
34	7	2	71	2	2	108	2	2
35	8	2	72	5	2	109	0	2
36	11	2	73	3	2			
37	6	2	74	2	2			

TABLE V. NUMBER OF FAULTS CLASSIFICATION

Faults Class	**Number of Faults**	**Number of Instances**		
		DS1	***DS2***	***DS3***
A	0-4	30	76	57
B	5-9	5	23	33
C	10-14	5	4	18
D	15-19	2	3	1
E	More than 20	4	5	0

TABLE VI. THE CONFUSION MATRIX

Predicted	**Actual**	
	Class X	*Class Y*
Class X	TP	FP
Class Y	FN	TN

B. Accuracy

Accuracy (ACC) is the proportion of true results (both TP and TN) among the total number of examined instances. The best accuracy is 1, whereas the worst accuracy is 0. ACC can be computed by using the following formula:

$$ACC = (TP + TN) / (TP + TN+ FP + FN) \quad (1)$$

C. Precision (Positive Predictive Value)

Precision is calculated as the number of correct positive predictions divided by the total number of positive predictions. The best precision is 1, whereas the worst is 0 and it can be calculated as:

$$Precision = TP / (TP + FP) \quad (2)$$

D. Recall (True Positive Rate or Sensitivity)

Recall is calculated as the number of positive predictions divided by the total number of positives. The best recall is 1, whereas the worst is 0. Generally, Recall is calculated by the following formula:

$$Recall = TP / (TP + FN) \quad (3)$$

E. F-measure

F-measure is defined as the weighted harmonic mean of precision and recall. Usually, it is used to combine the Recall and Precision measures in one measure in order to compare different ML algorithms with each other. F-measure formula is given by:

$$F\text{- measure}= (2^* Recall * Precision)/(Recall + Precision) \quad (4)$$

F. Root-Mean-Square Error (RMSE)

RMSE is a measure for evaluating the performance of a prediction model. The idea herein is to measure the difference between the predicted and the actual values. If the actual value is X and the predicted value is XP then RMSE is calculated as follows:

$$RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (Xi - XPi)^2} \quad (5)$$

V. EXPERIMENTAL RESULTS

This study used WEKA 3.6.9, a machine learning tool, to evaluate three ML algorithms (NB, DT and ANNs) in software bug prediction problem. A cross validation (10 fold) is used for each dataset.

The accuracy of NB, DT and ANNs classifiers for the three datasets are shown in Table VI. As shown in Table VI, the three ML algorithms achieved a high accuracy rate. The average value for the accuracy rate in all datasets for the three classifiers is over 93% on average. However, the lowest value appears for NB algorithm in the DS1 dataset. We believe this is because the dataset is small and NB algorithm needs a bigger dataset in order to achieve a higher accuracy value. Therefore, NB got a higher accuracy rate in DS2 and DS3 datasets, which they are relatively bigger than the DS1 dataset.

TABLE VII. ACCURACY MEASURE FOR THE THREE ML ALGORITHMS OVER DATASETS

Datasets	**NB**	**DT**	**ANNs**
DS1	0.898	0.951	0.938
DS2	0.950	0.972	0.954
DS3	0.954	0.990	0.963
Average	**0.934**	**0.971**	**0.951**

TABLE VIII. PRECISION MEASURE FOR THE THREE ML ALGORITHMS OVER DATASETS

Datasets	NB	DT	ANNs
DS1	0.956	1	1
DS2	0.989	0.990	0.981
DS3	0.990	1	0.990
Average	**0.978**	**0.996**	**0.990**

TABLE IX. RECALL MEASURE FOR THE THREE ML ALGORITHMS OVER DATASETS

Datasets	NB	DT	ANNs
DS1	1	1	1
DS2	0.905	1	0.990
DS3	0.972	1	0.981
Average	**0.959**	**1**	**0.990**

The precision measures for applying NB, DT and ANNs classifiers on DS1, DS2 and DS3 datasets are shown in Table VII. Results show that three ML algorithms can be used for bug prediction effectively with a good precision rate. The average precision values for all classifiers in the three datasets are more than 97%.

The third evaluation measure is the recall measure. Table VIII shows the recall values for the three classifiers on the three datasets. Also, herein the ML algorithms achieved a good recall value. The best recall value was achieved by DT classifier, which is 100% in all datasets. On the other hand, the average recall values for ANNs and NB algorithms are 99% and 96%, respectively.

In order to compare the three classifiers with respect to recall and precision measures, we used the F-measure value. Fig. 1 shows the F-measure values for the used ML algorithms in the three datasets. As shown the figure, DT has the highest F-measure value in all datasets followed by ANNs, then NB classifiers.

Finally, to evaluate the ML algorithms with other approaches, we calculated the RMSE value. The work in [2] proposed a linear Auto Regression (AR) model to predict the accumulative number of software faults using historical measured faults. They evaluated their approach with the POWM model [20] based on the RMSE measure. The evaluation process was done on the same datasets we are using in this study.

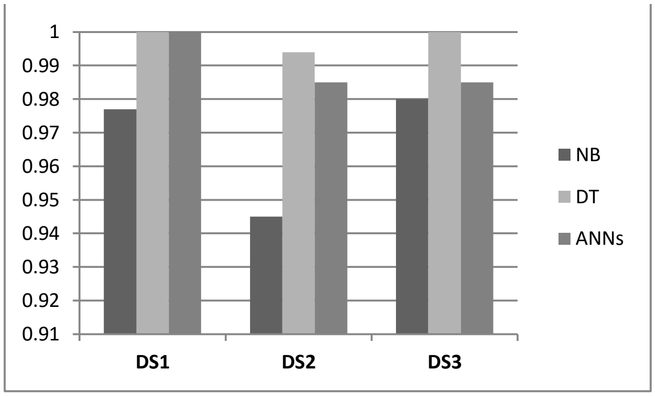

Fig. 1. F-measure values for the used ML algorithms in the three datasets.

TABLE X. RMSE VALUES FOR THE THREE ML ALGORITHMS, AR MODEL, AND POWM MODEL

Datasets	Machine Learning Algorithms			Approaches Presented in [2]	
	NB	***DT***	***ANNs***	***AR Model***	***POWM Model***
DS1	0.163	0.082	0.151	4.096	14.060
DS2	0.199	0.104	0.130	0.687	150.075
DS3	0.120	0.062	0.162	3.567	152.969

Table IX presents the RMSE measure for the used ML algorithms, as well as, AR and POWM models over the three datasets. The results show that NB, DT, and ANNs classifiers have better values than AR and POWM models. The average RMSE value for all ML classifiers in the three datasets is 0.130, while the average RMSE values for AR and POWM models are 2.783 and 105.701, respectively.

VI. CONCLUSIONS AND FUTURE WORK

Software bug prediction is a technique in which a prediction model is created in order to predict the future software faults based on historical data. Various approaches have been proposed using different datasets, different metrics and different performance measures. This paper evaluated the using of machine learning algorithms in software bug prediction problem. Three machine learning techniques have been used, which are NB, DT and ANNs.

The evaluation process is implemented using three real testing/debugging datasets. Experimental results are collected based on accuracy, precision, recall, F-measure, and RMSE measures. Results reveal that the ML techniques are efficient approaches to predict the future software bugs. The comparison results showed that the DT classifier has the best results over the others. Moreover, experimental results showed that using ML approach provides a better performance for the prediction model than other approaches, such as linear AR and POWM model.

As a future work, we may involve other ML techniques and provide an extensive comparison among them. Furthermore, adding more software metrics in the learning process is one possible approach to increase the accuracy of the prediction model.

REFERENCES

[1] Y. Tohman, K. Tokunaga, S. Nagase, and M. Y., “Structural approach to the estimation of the number of residual software faults based on the hyper-geometric districution model,” IEEE Trans. on Software Engineering, pp. 345–355, 1989.

[2] A. Sheta and D. Rine, “Modeling Incremental Faults of Software Testing Process Using AR Models ”, the Proceeding of 4th International Multi-Conferences on Computer Science and Information Technology (CSIT 2006), Amman, Jordan. Vol. 3. 2006.

[3] D. Sharma and P. Chandra, "Software Fault Prediction Using Machine-Learning Techniques," Smart Computing and Informatics. Springer, Singapore, 2018. 541-549.

[4] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," Applied Soft Computing 21, (2014): 286-297

[5] Malhotra, Ruchika. "A systematic review of machine learning techniques for software fault prediction." Applied Soft Computing 27 (2015): 504-518.

[6] D'Ambros, Marco, Michele Lanza, and Romain Robbes. "An extensive comparison of bug prediction approaches." Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. IEEE, 2010.

[7] Gupta, Dharmendra Lal, and Kavita Saxena. "Software bug prediction using object-oriented metrics." *Sādhanā* (2017): 1-15..

[8] M. M. Rosli, N. H. I. Teo, N. S. M. Yusop and N. S. Moham, "The Design of a Software Fault Prone Application Using Evolutionary Algorithm," IEEE Conference on Open Systems, 2011.

[9] T. Gyimothy, R. Ferenc and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," IEEE Transactions On Software Engineering, 2005.

[10] Singh, Praman Deep, and Anuradha Chug. "Software defect prediction analysis using machine learning algorithms." 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, IEEE, 2017.

[11] M. C. Prasad, L. Florence and A. Arya, "A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques," International Journal of Database Theory and Application, pp. 179-190, 2015.

[12] Okutan, Ahmet, and Olcay Taner Yıldız. "Software defect prediction using Bayesian networks." Empirical Software Engineering 19.1 (2014): 154-181.

[13] Bavisi, Shrey, Jash Mehta, and Lynette Lopes. "A Comparative Study of Different Data Mining Algorithms." International Journal of Current Engineering and Technology 4.5 (2014).

[14] Y. Singh, A. Kaur and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," Software Qual J, p. 3–35, 2010.

[15] Malhotra, Ruchika, and Yogesh Singh. "On the applicability of machine learning techniques for object oriented software fault prediction." Software Engineering: An International Journal 1.1 (2011): 24-37.

[16] A.TosunMisirli, A. se Ba¸ S.Bener,“A Mapping Study on Bayesian Networks for Software Quality Prediction”, Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, (2014).

[17] T. Angel Thankachan1, K. Raimond2, “A Survey on Classification and Rule Extraction Techniques for Data mining”,IOSR Journal of Computer Engineering ,vol. 8, no. 5,(2013), pp. 75-78.

[18] T. Minohara and Y. Tohma, “Parameter estimation of hyper-geometric distribution software reliability growth model by genetic algorithms”, in Proceedings of the 6th International Symposium on Software Reliability Engineering, pp. 324–329, 1995.

[19] Olsen, David L. and Delen, “ Advanced Data Mining Techniques ”, Springer, 1st edition, page 138, ISBN 3-540-76016-1, Feb 2008.

[20] L. H. Crow, “Reliability for complex repairable systems,” *Reliability and Biometry, SIAM*, pp. 379–410, 1974.