A Machine Learning based Analytical Approach for Envisaging Bugs

Dr. Anjali Munde
Amity College of Commerce and Finance
Amity University Uttar Pradesh
Noida, India

***Abstract*—A software imperfection is a shortcoming, virus, defect, mistake, breakdown or glitch in software that initiates it to establish an unsuitable or unanticipated result. The foremost hazardous components connected with a software imperfection that is not identified at an initial stage of software expansion are time, characteristic, expenditure, determination and wastage of resources. Faults appear in any stage of software expansion. Thriving software businesses emphasize on software excellence, predominantly in the early stage of the software advancement. In succession to disable this setback, investigators have formulated various bug estimation methodologies till now. Though, emerging vigorous bug estimation prototype is a demanding assignment and several practices have been anticipated in the text. This paper exhibits a software fault estimation prototype grounded on Machine Learning (ML) Algorithms. The simulation in the paper directs to envisage the existence or non-existence of a fault, employing machine learning classification models. Five supervised ML algorithms are utilized to envisage upcoming software defects established on historical information. The classifiers are Naïve Bayes (NB), Support Vector Machine (SVM), K- Nearest Neighbors (KNN), Decision Tree (DT) and Random Forest (RF). The assessment procedure indicated that ML algorithms can be manipulated efficiently with high accuracy rate. Moreover, an association measure is employed to evaluate the propositioned extrapolation model with other methods. The accumulated conclusions indicated that the ML methodology has an improved functioning.**

Keywords*—Software bug prediction; prediction model; data mining; machine learning; Naïve Bayes (NB); support vector machine (SVM); k-nearest neighbors (KNN); decision tree (DT); random forest (RF); python programming

I. Introduction

From the time of establishment of software expansion, defect restoration is studied as the most monotonous tasks, primarily for its in-built vagueness. Furthermore, the procedure of repairing bugs is gradual. The procedure of bug-restoration has a chief involvement in the software advancement. In order to lessen the concern of fault correction, bug estimation is examined significantly by the investigators. Numerous machine learning directed estimation prototypes are constructed and verified on several arguments.

The continuation of software faults influences considerably on software consistency, feature and upholding expense. Attaining errorless software is laborious, when the software utilized meticulously as largely there are unknown defects. Furthermore, extending software fault estimation prototype which can estimate the imperfect components in an initial stage is an actual test.

Contemporary developments rotate about the information that defects can be envisaged, widely beforehand they are identified. Significant corpuses of preceding fault information are fundamental to be proficient to envisage defects with sufficient precision. Software analytics has initiated continuous opportunities for tapping data analytics and rationalizing to enhance the feature of software. Functional analytics applies the outcomes of the software evaluation as real time data, to create valuable extrapolations.

Software defect estimation is an indispensable action in software expansion as envisaging the defects components earlier to software implementation attains the operator contentment and corrects the complete software functioning. Besides, envisaging the software fault initially increases software alteration to distinctive situations and enlarges the resource consumption.

Several methods are recommended to undertake Software fault estimation obstruction. The utmost comprehended procedures are Machine Learning procedures. Machine learning is effectively employed to build extrapolations in numerous database. Provided the enormous amount of fault database accessible currently, envisaging the occurrence of faults can be completed employing several machine learning procedures.

The application of machine learning to establish an exclusively mechanised technique of determining the act to be acquired by a business when a fault is testified was initially propositioned through Cubranic and Murphy [1]. The technique implemented helps text classification to envisage defect rigorousness. This technique functions accurately on 30% of the defects testified to creators. Sharma, Sharma and Gujral [2] apply feature selection to enhance the precision of the fault estimation prototype.

In this communication, supervised Machine Learning (ML) classifiers are employed to assess the ML potentials in Software fault estimation. The analysis examined Naïve Bayes (NB), Support Vector Machine (SVM), K- Nearest Neighbors (KNN), Decision Tree (DT) and Random Forest (RF) classifier. The considered ML classifiers are directed to three distinctive database acquired from [3] and [4] mechanisms.

Further, the manuscript evaluated among Naïve Bayes (NB) classifier, Support Vector Machine (SVM) classifier, K-

Nearest Neighbors (KNN) classifier, Decision Tree (DT) classifier and Random Forest (RF) classifier and compared then on the basis of distinct assessment quantities for instance accuracy, precision, recall, F-measures and the ROC curves of the classifiers.

The manuscript is structured as per the following sequence. An examination of the associated work in Software fault estimation is exhibited in the Literature Review. An outline of the designated ML algorithms is exhibited in the Proposed Model. The database and the assessment technique is explained in the Evaluation Methodology. Investigational outcomes are depicted in Results accompanied by inferences and future works.

II. Literature Review

Formerly, various efforts in the subjects of fault estimation have been achieved. Peng He et al. performed a practical analysis on software fault estimation with a basic metric set [5]. Investigation has been performed on 34 announcements of 10 open source assignments accessible at PROMISE repository. The outcome signifies the outcome of uppermost-k metrics or minimum metric subset gives satisfactory result in comparison with standard forecasters.

Anuradha Chug et al. [6] employed three supervised and unsupervised learning algorithms for envisaging faults in software. NASA MDP database were administered by utilizing Weka tool. Various quantities such as recall and f-measure were applied to estimate the functioning of classification and clustering algorithms. Through examining distinct classification algorithms Random Forest has the maximum accuracy of MC1 database and gives maximum rate in recall, f-measure and receiver operating characteristic [ROC] curve and it specifies least amount of root mean square errors in all conditions. In an unsupervised algorithm k-means gave the smallest amount of inaccurate clustered examples and it considers least period for envisaging defects. Hammouri, A. et al [7] proposed software defect estimation prototype established on Machine Learning Techniques to envisage impending software defects created on past database and exhibited that Machine Learning techniques can be applied successfully with high precision.

Logan Perreault et al. [8] employed classification algorithm for instance naïve bayes, neural networks, support vector machine, linear regression, K-nearest neighbor to discover and envisage faults. The investigators manipulated NASA and tera PROMISE database. To compute the accomplishment, they tapped accuracy and f1 measure with noticeably distinct metrics.

R. Malhotra in [9] exhibited a valuable methodical evaluation for software fault estimation procedures applying Machine Learning. The article encompassed an evaluation of all the findings concerning the interval of 1991 and 2013, examined the Machine Learning methods for software fault estimation prototypes, and evaluated their functioning, matched among Machine Learning and statistic methods, evaluated among distinct Machine Learning methods and reviewed the power and the limitation of the Machine Learning methods.

Singh and Chug [10] examined widespread Machine Learning algorithms tapped for software fault estimation. The analysis exhibited significant outcomes comprising that the Artificial Neural Network has least inaccuracy amount, but the linear classifier is advanced than auxiliary algorithms in term of fault estimation precision.

Malhotra and Singh [11] indicated that the Area Under Curve is constructive metric and utilised to envisage the defects in initial stages of software expansion and to increase the validity of Machine Learning methods.

This article examines established machine learning techniques Naïve Bayes (NB), Support Vector Machine (SVM), K- Nearest Neighbors (KNN), Decision Tree (DT) and Random Forest (RF). The communication estimates the Machine Learning classifiers by means of different performance quantities. Three known database are employed to assess the Machine Learning classifiers.

Alternatively, maximum cited assignments examined new Machine Learning techniques and distinct database. Few of the earlier investigations primarily concentrated on the metrics that generate the Software fault estimation as feasible as imaginable, though earlier investigations suggested distinctive approaches to estimate software faults in place of Machine Learning methods.

III. Proposed Model

The research directs to examine and assess supervised Machine Learning algorithms. The investigation exhibits the performance correctness and competency of the Machine Learning algorithms in software fault estimation and postulates a comparative study of the designated Machine Learning algorithms.

The supervised machine learning algorithms attempt to create an extrapolating function through deducing associations and needs among the identified feed in and outturn of the categorized training data, thus we can envisage the outturn amounts for recent feed in data created on the resulting extrapolating function. Subsequently are encapsulated explanations of the designated supervised Machine Learning algorithms:

- Naïve Bayes (NB): Naïve Bayes classifier functions on the theory of probability. The notion of naïve bayes classifier is established on the effort of Thomas Bayes (1702-1761) of Bayes Theorem for conditional probability. Naïve Baye’s Classifier performs on the notion of baye’s theorem through a naïve theory that an existence of a specific feature in a class is entirely discrete to the existence of additional features.

- Support Vector Machine (SVM): SVM is most widespread supervised machine learning technique which is equivalently tapped for classification and regression, however SVM is typically utilised for classification. The notion of SVM is to obtain a hyperplane that categorizes the training data points in order to obtain marked classes. The feed in of SVM is the training data and it functions the training sample feature to envisage category of test feature.

- K Nearest Neighbour (KNN): KNN algorithm well - known by K-Nearest Neighbours Algorithm is tapped to elucidate the difficulties of classification together with regression. The theory of algorithm is primarily established upon feature comparison in two of them, classification and regression. KNN classifier is distinct from previous probabilistic classifiers as the simulation encompasses a discovering phase of calculating probabilities from a training experiment and employ them for impending estimation of a test experiment. In probability established prototype when the prototype is proficient the training experiment could be dropped and classification is completed by means of the calculated probabilities.

- Decision Tree (DT): DT is a familiar investigation technique utilised in data mining. Decision Tree signifies a hierarchal and extrapolative prototype that utilises the elements examination as branches to access the elements target amount in the leaf. Decision Tree is a tree with decision nodes, that have several branches and leaf nodes that characterise the conclusion.

- Random Forest (RF): Random Forest comprises of a substantial quantity of distinct decision trees that function as an ensemble. Individual tree in the random forest separate out a class estimation. The class that has the highest votes turns out to be prototypes estimation. A big quantity of comparatively disjointed prototypes (trees) functioning as a group will outshine any of the specific prototypes.

IV. Evaluation Methodology

The database used in the study are three different databases, specifically DB1, DB2 and DB3. All databases comprise of two measures; the amount of defects (Bi) and the amount of test workers (Wi) for respective day (Ti) in a section of software launches period. The DB1 database has 46 quantities that were included in the examining procedure exhibited in [4]. DB2, captured from [4], computed a technique where defects for the period of 111 consecutive days of examining the software technique. DB3 includes 109 quantities. DB3 is established in [3], that comprises actual calculated records for a restoration plan of a real time control utilization exhibited in [12]. Tables I to III show DB1, DB2 and DB3, respectively.

TABLE I. The First Software Defects Database

The first software defects database	**DB1**		
	Ti	***Bi***	***Wi***
	1	2	75
	2	0	31
	3	30	63
	4	13	128
	5	13	122
	6	3	27
	7	17	136
	8	2	49
	9	2	26
	10	20	102
	11	13	53
	12	3	26
	13	3	78
	14	4	48
	15	4	75
	16	0	14
	17	0	4
	18	0	14
	19	0	22
	20	0	5
	21	0	9
	22	30	33
	23	15	118
	24	2	8
	25	1	15
	26	7	31
	27	0	1
	28	22	57
	29	2	27
	30	5	35
	31	12	26
	32	14	36
	33	5	28
	34	2	22
	35	0	4
	36	7	8
	37	3	5
	38	0	27
	39	0	6
	40	0	6
	41	0	4
	42	5	0
	43	2	6
	44	3	5
	45	0	8
	46	0	2

TABLE II. The Second Software Defects Database

The second software defects database	**DB2**		
	Ti	***Bi***	***Wi***
	1	5	4
	2	5	4
	3	5	4

The second software defects database	**DB2**		
	Ti	***Bi***	***Wi***
	4	5	4
	5	6	4
	6	8	5
	7	2	5
	8	7	5
	9	4	5
	10	2	5
	11	31	5
	12	4	5
	13	24	5
	14	49	5
	15	14	5
	16	12	5
	17	8	5
	18	9	5
	19	4	5
	20	7	5
	21	6	5
	22	9	5
	23	4	5
	24	4	5
	25	2	5
	26	4	5
	27	3	5
	28	9	6
	29	2	6
	30	5	6
	31	4	6
	32	1	6
	33	4	6
	34	3	6
	35	6	6
	36	13	6
	37	19	8
	38	15	8
	39	7	8
	40	15	8
	41	21	8
	42	8	8
	43	6	8
	44	20	8
	45	10	8
	46	3	8
	47	3	8
	48	8	4
	49	5	4
	50	1	4
	51	2	4

The second software defects database	**DB2**		
	Ti	***Bi***	***Wi***
	52	2	4
	53	2	4
	54	7	4
	55	2	4
	56	0	4
	57	2	4
	58	3	4
	59	2	4
	60	7	4
	61	3	4
	62	0	4
	63	1	4
	64	0	4
	65	1	4
	66	0	4
	67	0	4
	68	1	3
	69	1	3
	70	0	3
	71	0	3
	72	1	3
	73	1	4
	74	0	4
	75	0	4
	76	0	4
	77	1	4
	78	2	2
	79	0	2
	80	1	2
	81	0	2
	82	0	2
	83	0	2
	84	0	2
	85	0	2
	86	0	2
	87	2	2
	88	0	2
	89	0	2
	90	0	2
	91	0	2
	92	0	2
	93	0	2
	94	0	2
	95	0	2
	96	1	2
	97	0	2
	98	0	2
	99	0	2

The second software defects database	**DB2**		
	Ti	***Bi***	***Wi***
	100	1	2
	101	0	1
	102	0	1
	103	1	1
	104	2	1
	105	0	1
	106	1	2
	107	0	2
	108	0	1
	109	1	1
	110	0	1
	111	1	1

TABLE III. THE THIRD SOFTWARE DEFECTS DATABASE

The Third Software Defects Database	**DS3**		
	Ti	***Bi***	***Wi***
	1	4	1
	2	0	1
	3	7	1
	4	10	1
	5	13	1
	6	8	1
	7	13	1
	8	4	1
	9	7	1
	10	8	1
	11	1	1
	12	6	1
	13	13	1
	14	7	1
	15	9	1
	16	8	2
	17	5	2
	18	10	2
	19	7	2
	20	11	2
	21	5	2
	22	8	2
	23	13	2
	24	9	2
	25	7	2
	26	7	2
	27	5	2
	28	7	2
	29	6	1
	30	6	1
	31	4	1
	32	12	2
	33	6	2
	34	7	2
	35	8	2
	36	11	2
	37	6	2
	38	9	2
	39	7	2
	40	12	2
	41	12	2
	42	15	2
	43	14	2
	44	7	2
	45	9	2
	46	11	2
	47	5	2
	48	7	2
	49	7	2
	50	14	2
	51	13	2
	52	14	2
	53	11	2
	54	2	1
	55	4	1
	56	4	2
	57	3	2
	58	6	2
	59	6	2
	60	2	2
	61	0	1
	62	0	1
	63	3	1
	64	0	1
	65	4	1
	66	0	1
	67	1	1
	68	2	1
	69	0	2
	70	1	2
	71	2	2
	72	5	2
	73	3	2
	74	2	2
	75	1	2
	76	11	2
	77	1	2
	78	0	2

The Third Software Defects Database	**DS3**		
	Ti	***Bi***	***Wi***
	79	2	2
	80	2	2
	81	4	2
	82	1	2
	83	0	2
	84	4	2
	85	1	1
	86	1	1
	87	0	1
	88	2	3
	89	0	1
	90	0	2
	91	1	1
	92	1	1
	93	0	1
	94	0	2
	95	0	1
	96	0	1
	97	1	2
	98	0	1
	99	1	1
	100	0	1
	101	0	1
	102	0	2
	103	0	1
	104	2	1
	105	0	1
	106	1	2
	107	0	2
	108	2	2
	109	0	2

The database was subjected to pre-treatment through a recommended clustering method. The recommended clustering method indicates the data with class labels. The labels are fixed to categorize the amount of defects into six distinct classes; A, B, C, D, E and F (Table IV).

TABLE IV. AMOUNT OF EVERY CLASS AND QUANTITY OF OCCURENCES

Amount of Every Class and Quantity of Occurrences				
Fault Class	***Number of Faults***	***DB1***	***DB2***	***DB3***
A	0-4	30	77	57
B	5-9	5	22	33
C	10-14	5	4	18
D	15-19	2	3	1
E	20-24	2	3	0
F	More than 25	2	2	0

To estimate the functioning of utilising Machine Learning algorithms in software fault extrapolation, we tapped an array of prominent quantities [13] established on the created confusion matrices. The subsequent subdivisions explain the confusion matrix and the tapped estimation quantities.

a) Confusion Matrix: The confusion matrix is an explicit table employed to determine the functioning of Machine Learning algorithms. Fig. 1 to 6 exhibits an illustration of a standard confusion matrix. Every row of the matrix signifies the occurrences in an actual class, although every column signifies the occurrences in a forecasted class. Confusion matrix recapitulates the outcomes of the examining algorithm and specifies a description of the amount of True Positive (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

Confusion Matrix for the Training Data - Decision Tree.

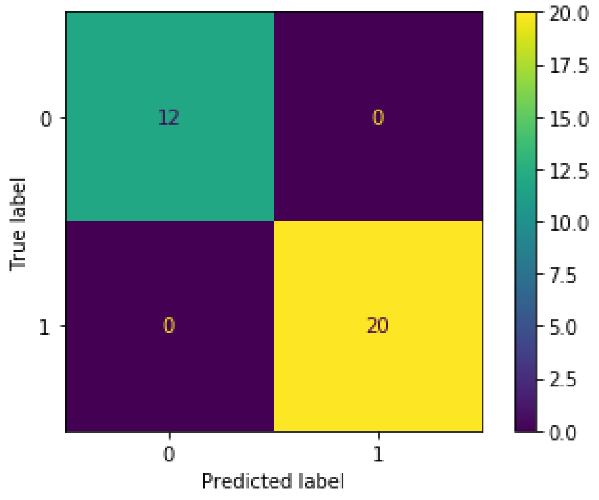

Fig. 1. Confusion Matrix of Training Data for DB1.

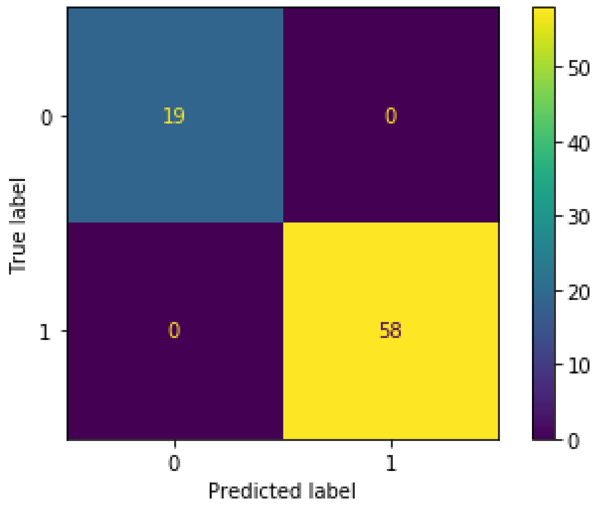

Fig. 2. Confusion Matrix of Training Data for DB2.

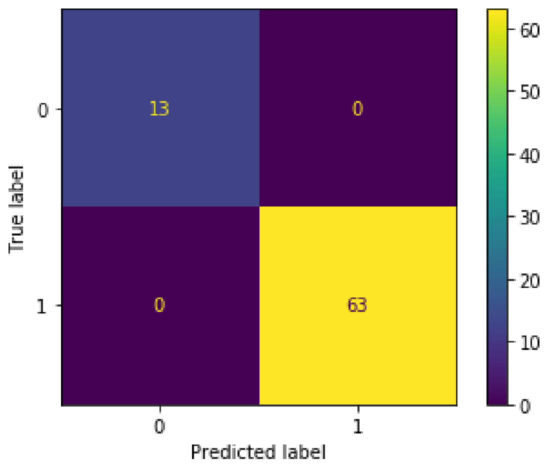

Fig. 3. Confusion Matrix of Training Data for DB3.

Confusion Matrix for the Testing Data - Decision Tree.

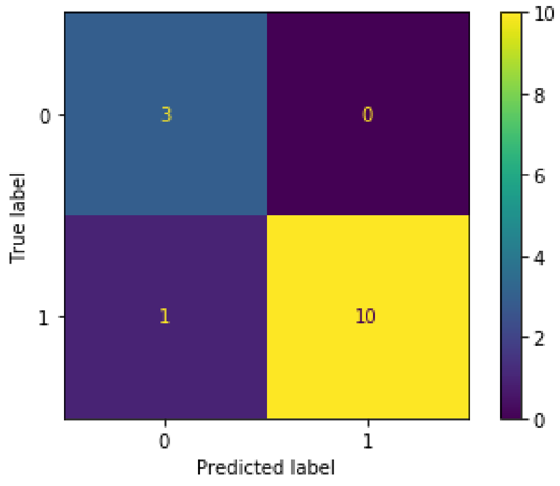

Fig. 4. Confusion Matrix of Testing Data for DB1.

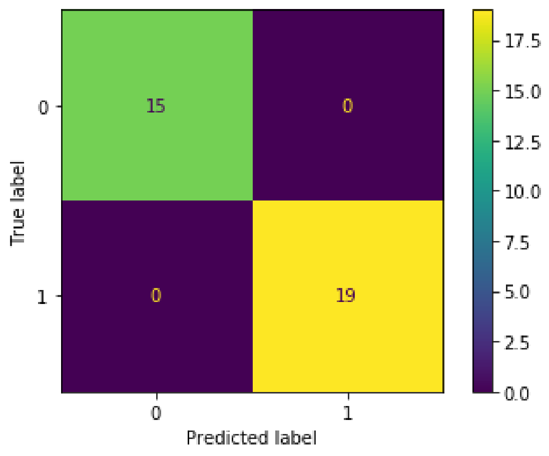

Fig. 5. Confusion Matrix of Testing Data for DB2.

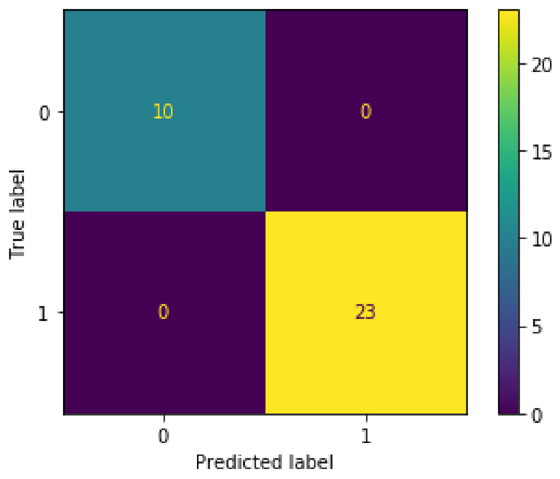

Fig. 6. Confusion Matrix of Testing Data for DB3.

b) Accuracy: Accuracy is the quantity of accurate outcomes between the total amount of inspected occurrences. The highest accuracy is one, while the poorest accuracy is zero. Accuracy could be calculated through the subsequent rule (Table V):

ACC = (TP + TN) / (TP + TN+ FP + FN)

TABLE V. ACCURACY FOR THE DATABASES

Accuracy for the Databases					
Database	***NB***	***SVM***	***KNN***	***DT***	***RF***
DB1	1	0.71	0.79	0.93	0.93
DB2	1	0.85	0.79	1	1
DB3	1	0.70	0.70	1	1
Average	1	0.75	0.76	0.98	0.98

c) Precision: Precision is computed as the amount of true positive extrapolations divided with the total amount of positive extrapolations. The highest precision is one, while the poorest is zero and could be computed through (Table VI):

Precision = TP / (TP + FP)

d) Recall: Recall is computed as the amount of positive extrapolations divided with the total amount of positives. The highest recall is one, while the poorest is zero. Recall is evaluated through the subsequent rule (Table VII):

Recall = TP / (TP + FN)

TABLE VI. PRECISION FOR THE DATABASES

Precision for the Databases					
Database	***NB***	***SVM***	***KNN***	***DT***	***RF***
DB1	1	1	1	1	1
DB2	1	0.85	0.77	1	1
DB3	1	0.70	0.78	1	1
Average	1	0.85	0.85	1	1

TABLE VII. RECALL FOR THE DATABASES

Recall for the Databases					
Database	***NB***	***SVM***	***KNN***	***DT***	***RF***
DB1	1	0.64	0.73	0.91	0.91
DB2	1	0.89	0.89	1	1
DB3	1	1	0.78	1	1
Average	1	0.84	0.80	0.97	0.97

e) F-measure: F-measure is described by way of the weighted harmonic mean of precision and recall. Generally, it is tapped to join the Recall and Precision quantities in one quantity so as to evaluate distinct Machine Learning algorithms among each other. F-measure rule is evaluated through the subsequent rule (Table VIII):

F- measure= (2* Recall * Precision)/(Recall + Precision)

TABLE VIII. F-MEASURE FOR THE DATABASES

F-Measure for the Databases					
Database	***NB***	***SVM***	***KNN***	***DT***	***RF***
DB1	1	0.78	0.84	0.95	0.95
DB2	1	0.87	0.83	1	1
DB3	1	0.82	0.78	1	1
Average	1	0.82	0.82	0.98	0.98

f) Root-Mean-Square Error (RMSE): RMSE is a quantity for assessing the functioning of an extrapolation prototype. The perception is to compute the variation among the envisaged and the definite estimates. If the definite estimate is X and the envisaged estimate is XP then RMSE is computed by the subsequent formula:

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

g) Area Under Curve(AUC): AUC exemplifies the probability that the classifier would rank an arbitrarily selected positive instance greater than an arbitrarily selected negative instance. The AUC is established on a chart of the false positive value with the true positive value. The highest value is one signifies that 100% estimation of the model is accurate, while the poorest is zero signifies that 100% estimation of the model is inaccurate. Fig. 7 to 12 exhibits an illustration of the Area Under Curve.

h) Receiver Operating Characteristic (ROC): ROC Curve is an outstanding technique of calculating the functioning of a Classification prototype. The True Positive value is plot with False Positive value for the probabilities of a classifier estimations.

AUC and ROC for the Training Data - Decision Tree

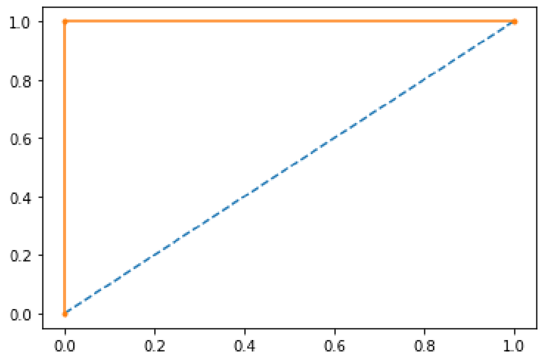

Fig. 7. AUC of Training Data for DB1.

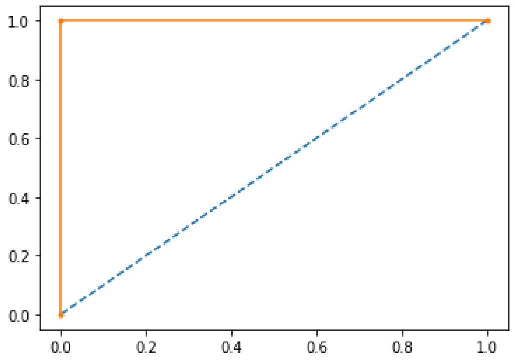

Fig. 8. AUC of Training Data for DB2.

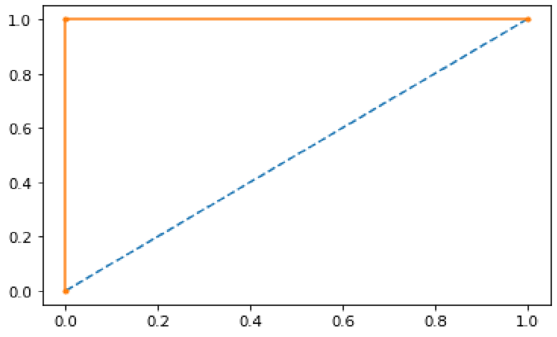

Fig. 9. AUC of Training Data for DB3.

AUC and ROC for the Testing Data - Decision Tree.

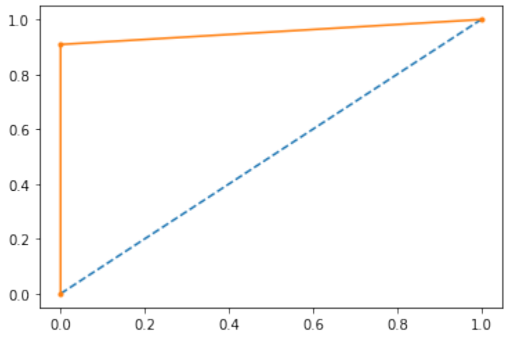

Fig. 10. AUC of Testing Data for DB1.

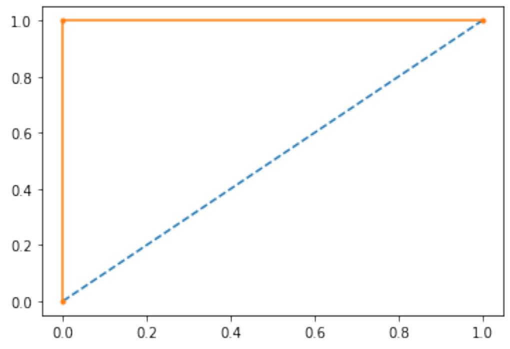

Fig. 11. AUC of Testing Data for DB2.

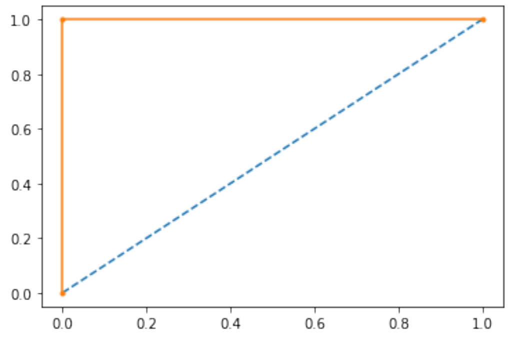

Fig. 12. AUC of Testing Data for DB3.

Conclusively, to assess the Machine Learning algorithms with additional methods, the RMSE value is estimated. The composition in [2] anticipated a Linear Regression (LR) prototype to envisage the increasing amount of software defects utilising past calculated defects. The assessment procedure was performed on the similar database that is used in this investigation. The lesser the RMSE amount, the reliable the model. Table IX exhibits the RMSE values for all the ML Algorithms and LR models.

TABLE IX. RMSE VALUES FOR THE ML ALGORITHMS AND LR MODELS

RMSE Values for the ML Algorithms and LR Models						
Database	***NB***	***SVM***	***KNN***	***DT***	***RF***	***LR***
DB1	0.0	0.53	0.53	0.26	0.26	0.43
DB2	0.0	0.38	0.38	0.0	0.0	0.38
DB3	0.0	0.55	0.55	0.0	0.0	0.36

V. RESULTS

This inquiry utilised Jupyter Notebook, Python as Machine Learning tool, to assess five Machine Learning Algorithms Naïve Bayes (NB), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree (DT) and Random Forest (RF)) in software default estimation.

The accuracy of Naïve Bayes (NB), Support Vector Machine (SVM), K- Nearest Neighbors (KNN), Decision Tree (DT) and Random Forest (RF) classifiers for the three database are presented in Table VI. As exhibited in Table VI, the five Machine Learning algorithms attained a high accuracy value.

The typical estimate for the accuracy value in all database for the five classifiers is over 75% on average. Though, the lowermost estimate emerges for SVM and KNN algorithm in the DS3 database. This is for the reason that the database does not have greater than 20 defects and SVM and KNN algorithm requires a significant quantity of defects so as to attain a better accuracy rate. Thus, SVM and KNN got a greater accuracy value in DS2 database that are comparatively larger than the DS1 and DS3 database.

The precision measures for employing NB, SVM, KNN, DT and RFs classifiers on DS1, DS2 and DS3 database are exhibited in Table VII. Outcomes indicate that the five Machine Learning algorithms can be utilised for defect extrapolation successfully with a right precision value. The typical precision rates for every classifier in the three database are greater than 85%.

The next assessment quantity is the amount of recall. Table VIII exhibits the recall rates for the five classifiers on the three database. Correspondingly, the Machine Learning algorithms attained a suitable recall rate. The highest recall rate was attained by NB classifier that is 100% in all database. Whereas, the typical recall rates for SVM, KNN, DT and RM algorithms are 84%, 80%, 97% and 97%, correspondingly.

Further, to evaluate the five classifiers concerning recall and precision quantities, we employed the F-measure rate. Table exhibits the F-measure rates for the utilised Machine Learning algorithms in the three database. As presented in the table, NB has the maximum F-measure rate in all database trailed by DT and RF then SVM and KNN classifiers.

The outcomes represent that NB, DT and RF classifiers have improved rates than LR models. The typical RMSE amount for all Machine Learning classifiers in the three database is 0.28, whereas the typical RMSE estimates for LR model is 0.39.

VI. CONCLUSIONS AND FUTURE WORK

Software fault estimation is a procedure in which an extrapolation prototype is generated so as to envisage the anticipated software defects created on past data. Numerous methodologies have been propositioned utilising distinct database, distinct metrics and distinct functioning quantities. This article assessed the application of Machine Learning Algorithms in software defect estimation. Five machine learning methods have been employed, Naïve Bayes (NB), Support Vector Machine (SVM), K- Nearest Neighbors (KNN), Decision Tree (DT) and Random Forest (RF). The assessment procedure is applied utilising three database. Investigational outcomes are accumulated built on accuracy, precision, recall, F-measure, and RMSE quantities. Outcomes showed that the Machine Learning procedures are effective methods to envisage the impending software faults. The evaluation outcomes exhibited that the NB classifier has the greatest outcomes in comparison to others. Furthermore, investigational outcomes presented that employing Machine Learning method imparts an improved functioning for the estimation prototype in comparison to other methods, such as LR model. For future scope, new Machine Learning procedures can be adopted and an extensive assessment between them can be performed. Moreover, inserting additional software metrics in the study procedure is a feasible method to foster the correctness of the estimation model.

References

[1] D. Cubranic and G.C. Murphy, "Automatic bug triage using text classification," Proceedings of Software Engineering and Knowledge Engineering, pp. 92–97, 2004.

[2] G. Sharma, S. Sharma and S. Gujral, "A Novel Way of Assessing Software Bug Severity Using Dictionary of Critical Terms;" Procedia Computer Science, vol 70, pp. 632–639, 2015.

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

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

[5] P. He., B. Li, X. Liu, J. Chen and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," Information and Software Technology, vol. 59, pp. 170-190, 2015.

[6] A. Chug and S. Dhall, "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm," Confluence 2013: The Next Generation Information Technology Summit, pp. 5-10, 2013.

[7] A. Hammouri, M. Hammad, M. Alnabhan and F. Alsarayrah, "Software Bug Prediction using Machine Learning Approach," International Journal of Advanced Computer Science and Applications, vol. 9(2), pp. 78-83, 2018.

[8] L. Perreault, S. Berardinelli, C. Izurieta and J. Sheppard, "Using Classifiers for Software Defect Detection," 26th International Conference on Software Engineering and Data Engineering, SEDE, 2017.

[9] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Applied Soft Computing, vol. 27, pp. 504-518, 2015.

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

[11] R. Malhotra and Y. Singh, "On the applicability of machine learning techniques for object oriented software fault prediction," Software Engineering: An International Journal, vol. 1(1), pp. 24-37, 2011.

[12] 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.

[13] Olsen, L. David and Delen, "Advanced Data Mining Techniques," Springer, 1st edition, pp. 138, ISBN 3-540-76016-1, Feb 2008.