International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

Advanced Security system in Cloud computing using Machine Learning

Dr. M. Aruna Safali

Associate Professor, CSE, Dhanekula Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India

ARTICLEINFO

Article History:

Accepted: 10 Feb 2023
Published: 28 Feb 2023

Publication Issue
Volume 10, Issue 1
January-February-2023

Page Number
261-271

ABSTRACT

Cloud computing (CC) is on-demand accessibility of network resources, especially data storage and processing power, without special and direct management by the users. CC recently has emerged as a set of public and private datacenters that offers the client a single platform across the Internet. Edge computing is an evolving computing paradigm that brings computation and information storage nearer to the end-users to improve response times and spare transmission capacity. Mobile CC (MCC) uses distributed computing to convey applications to cell phones. However, CC and edge computing have security challenges, including vulnerability for clients and association acknowledgment, that delay the rapid adoption of computing models. Machine learning (ML) is the investigation of computer algorithams that improve naturally through experience. In this review paper, we present an analysis of CC security threats, issues, and solutions that utilized one or several ML algorithms. We review different ML algorithms that are used to overcome the cloud security issues including supervised, unsupervised, semi-supervised, and reinforcement learning. Then, we compare the performance of each technique based on their features, advantages, and disadvantages. Moreover, we enlist future research directions to secure CC models.

Keywords: Cloud computing, Machine learning, supervised, unsupervised, semi-supervised, and reinforcement learning, AUC, ROC, TPR, FPR, KNN

I. INTRODUCTION

Reasonably bugged or defect free, delivered on time, meet the requirements or expectations, within a specified budget, and is maintainable are points which are referred as software quality parameters. Finding software fault from the system leads to improve quality of a software. There could be many reasons for occurrence of software fault. It is due to human mistakes, errors made by designer, programmer, incorrect data entry, documentation, communication failure, wrong code of lines.

To identify fault/defect/bug we need metrics that can measure the faults from the system. To examine software quality, software engineers have restricted resources and analysis tools for testing. The objective

Copyright: © the author(s), publisher and licensee Technoscience Academy. This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License, which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited

of testing is to find errors/bugs in the system. Best and successful test cases can be used to find even the undiscovered and uncovered errors with high probability[1].

Buggy modules are detected using Halstead, LoC and McCabes's attributes in many researches. True Positive rate (TPR), False Positive Rate (FPR), Precision, Confusion Matrix, Area Under Receiver Operating Curve (AUC-ROC) are used to measure performance of classifiers [2] [3]. According to research, AUC-ROC, AUC-PR are used to appraise the skewed data distribution in many cases.

II. RELATED WORK

To reduce rework and cost, the developers must find the faults in the early stages of development. The analysis report shows that a combination of machine learning techniques may produce better prediction models than the current methods. It may also reduce cost and rework of software development at the same time [4]. There are various machine learning algorithms for classification (supervised learning).

These families includes adaboost, decision trees, support vector machines (SVM), neural networks [5] and deep learning, random forest, k-nearest neighbors (KNN), extra trees, logistic regression and gradient boosting [6]. Machine learning techniques are being used in software fault prediction to assist testing and maintainability. Fault prediction approaches are explored by researchers in the literature [7].

Several researchers have also contributed in distributed software systems using machine learning. The key of software defect prediction is how to effectively analyse and use existing historical data for creating more precise classifiers [8]. The classification approaches often encounter certain difficulties including the issue of misclassification cost [9] and the class imbalance problem [1] [10].

Previous studies showed that, all the classifiers perform different in different scenario. Parameters like labelled, un-labelled, size, Training and test dataset split ratio and other factor affect the result to certain extent [11]. Investigators have previously explored and suggested different classification approaches. Naïve Bayes was one of the most widely-used approaches, but there are other alternatives. Logistic regression is another popular approach, as is decision tree classifiers, support vector machines (SVM) and ensemble techniques such as K-means clustering. Supervised learning doesn't use literal data; unlabelled data is used instead [7] [12].

Datasets are utilised from the public repositories which are available for research which includes source code changes, mail archives and version control. These datasets also have information like coupling between object classes, depth of inheritance, McCabe's Cyclomatic complexity, number of classes, interfaces, methods and other data also [11].

Data sets are generated from software repositories including defect tracking systems, source code changes, mail archives, data extraction and version control systems. Those data sets consist of instances, which can be software components, files, classes, functions and modules. Based on particular metrics like static code attributes [13] extracted from the software repositories, an instance is labelled as defective or defect-free. The collected data sets are then cleaned using preprocessing methods such as noise detection and reduction, data normalization, and attribute selection [14]. After that, the preprocessed data sets are used for building a defect prediction model that is to predict whether new instances contain defects or not.

The researcher evaluated traditional machine learning, deep learning based and hybrid learning based techniques. As a result, the research on just-in-time is valued. Deep learning and hybrid learning have produced numbers of state-of-art methods that can significantly improve prediction performance, aiming to predict defects of both cross-project and within-project [15]. In this study had used Cross-project defect prediction, which often reuse data from other projects. It works well when the data of training models is completely sufficient to meet the project demands. In this paper, he have modeled the outcomes using PROMISE dataset in five different modules and repositories: CM1, JM1, KC1, KC2, and PC1. We implemented the dataset using four different classifiers: Bayes network, Random forest, SVM, and the Deep Learning based on F-measure, making it more robust and outperform all the models available [12] [15]. However, current studies on software defect prediction require some degree of heterogeneity of metric values that does not always lead to accurate predictions.

In this paper bug prediction is done using deep representation and ensemble learning (BPDET) techniques [16]. Ensemble learning (Staked denoising auto-encoders) and deep Presented BPDET performed better for the most of used datasets compared to the AdaBoost, Bagging, Random forest, and Logistic Boost.

There are so many prediction approaches in the field of software engineering such as test effort, security and cost prediction. Since most of them do not have a stable model, software fault prediction has been studied in this paper based on different machine learning techniques such as decision trees, decision tables, random forest, neural network, Naïve Bayes and distinctive classifiers of artificial immune systems (AISs) such as artificial immune recognition system, CLONALG and Immunos [17].

III.RESEARCH METHODOLOGY

To check the performance of software fault prediction model, researcher select machine learning classifier like Random Prediction, Random Forest, Naïve Bayes, Logistic Regression, Decision Tree, Gradient Boosting, Support Vector Machine and K-Nearest Neighbor which are implemented on Datasets Like Ant, Camel, Ivy, Jedit, Log4j, Lucane, Poi, Xalan, Xerces with their Versions.

Researcher evaluated and examined each model and then the study of AUC-ROC has been done.

A. Experimental Datasets

To perform research Nine datasets with twenty eight versions are used. They are Camel, Ivy, Jedit, Log4J, Lucene, Poi, Xalan, Xereces, Prop. Table 1. Shows the dataset name, Versions, LOC, Total number of modules, total number of modules without defect, total number of modules with defects and percentage of defect modules.

B. Model Development

In this study, eight classification algorithms were considered for model development which has been mention inn Table-2. All these classifiers have been used benchmarking study 2, [5]. The Classifiers are Random Prediction, Random Forest, Naïve Bayes, Logistic Regression, Decision Tree, Gradient Boosting, Support Vector Machine and K-Nearest Neighbor

TABLE I
CLASSIFIER LIST USED FOR RESEARCH

S. N	Classifiers	Abbreviation
1	Random (chance) Prediction	RP
2	Random Forest	RF
3	Naïve Bayes	NB
4	Logistic Regression	LR
5	Decision Tree	DT
6	Gradient Boosting	GB
7	Support Vector Machine	SVM
8	K-Nearest Neighbor	KNN

C. Parameters for Evaluation

To evaluate model performance researcher use 2 parameters i.e. TPR and FPR

Sensitivity / True Positive Rate / Recall (TPR) is
Correctly classified Negative classes = Specificity

False Positive Rate (FPR) is
Correctly classified positive classes = False Negative Rate

FPR tells us what proportion of the negative class got incorrectly classified by the classifier.
Where,
Specificity / True Negative Rate is
Correctly classified Negative classes = Specificity

IV. MATERIAL AND METHODS

Machine learning techniques are being used in software fault prediction to assist testing and maintainability. Fault prediction approaches are explored by researchers in the literature [18]. Several researchers have also contributed in distributed software systems using machine learning [19]. Previous studies showed that, all the classifiers perform different in different scenario. Parameters like labelled, un-labelled, size, Training and test dataset split ratio [20] and other factor affect the result to certain extent.

Investigators have explored and suggested different classification methods, including Naïve Bayes, Logistic Regression, Decision Tree, Support Vector Machine, Ensemble Approaches, K-Means Clustering & Fuzzy Clustering [21]. Investigators use supervised learning to label the data; they do not use literal data.

To check the performance of software fault prediction model, researcher had selected Random Prediction, Random Forest, Naïve Bayes, Logistic Regression, Decision Tree, Gradient Boosting, Support Vector Machine and K-Nearest Neighbor which are implemented on Datasets viz Ant, Camel, Ivy, Jedit, Log4j, Lucane, Poi, Xalan, Xerces with their Versions.

Researcher evaluated and examined each model and then the study of AUC-ROC has been done according to the below proposed algorithm.

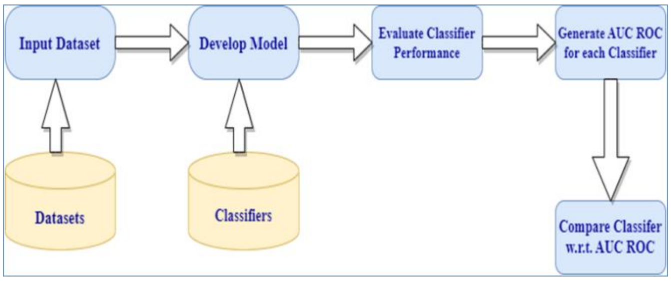

Input Dataset
Develop Model
Evaluate Classifier Performance
Generate AUC ROC for each Classifier
Datasets
Classifiers
Compare Classifer w.r.t. AUC ROC

Figure 1. Proposed Experimental method

Algorithm-1

Step 1: Read Dataset

Step 2: For each dataset - Repeat

Step 3: Apply Data Preprocessing

Step 4: Split Dataset into Training set and Testing Set

Step 5: Build Model, Apply Classification Model on Test Data.

[**Classification model used :** K Nearest Neighbor, SVM, Gradient Boosting, Decision Tree, Random Forest, Naïve Bayes, Logistic Regression]

Step 6: Finding Confusion Matrix

Step 7: Finding Values of TP,TN,FP,FN from Confusion Matrix

Step 8: Record TPR and FPR

Step 9: Find AUC-ROC values for dataset with respect to ML Classifier

Step 10: Repeat the steps 5 to 9

Step 11: Perform Comparative Analysis of dataset with respect to ML Classifier.

Step 12: Stop

V. RESULTS AND DISCUSSION

Results assessments are based on experiments done by researchers. AUC-ROC experimental results has been mention in Table II (A)(B)(C)(D)(E)(F)(G)(H)(I) with classifiers and datasets. In which, researchers calculated AUC-ROC for each model for all 28 datasets. *Table 3* shows AUC-ROC with the best score for applied models.

Table II(A),(B),(C),(D),(E) and (F) shows that Support Vector Machine (SVM) is more effective, which gives score 1 or approximately 1 followed by Gradient Boosting as second highest score.

TABLE III

(A) DATASET CAMEL V/S CLASSIFIERS

Datasets	Camel-1.0	Camel-1.2	Camel-1.4	Camel-1.6
Classifiers				
Random (chance) Prediction	0.500	0.500	0.500	0.500
Random Forest	0.787	0.711	0.750	0.710
Naïve Bayes	0.964	0.480	0.699	0.688
Logistic Regression	0.497	0.584	0.722	0.652
Decision Tree	0.515	0.585	0.591	0.724
Gradient Boosting	0.315	0.688	0.753	0.674
Support Vector Machine	1.000	0.721	0.237	0.711
K-Nearest Neighbor	0.636	0.617	0.688	0.655

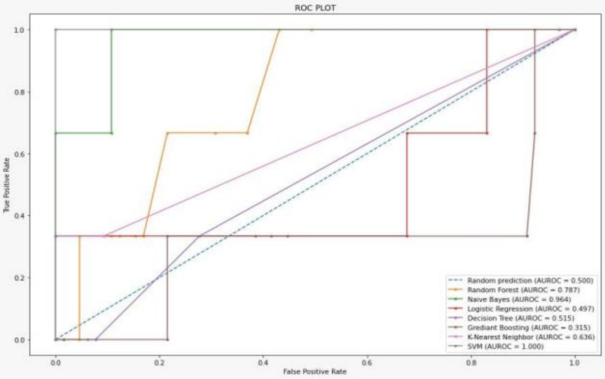


Figure 2. AUC-ROC graph for Camel Dataset

TABLE IIII

(B) DATASET IVY V/S CLASSIFIERS

Dataset-	**Ivy-2.0**
Classifiers	
Random (chance) Prediction	0.500
Random Forest	0.861
Naïve Bayes	0.769
Logistic Regression	0.930
Decision Tree	0.723
Grediant Boosting	0.826
Support Vector Machine	0.954
K-Nearest Neighbor	0.684

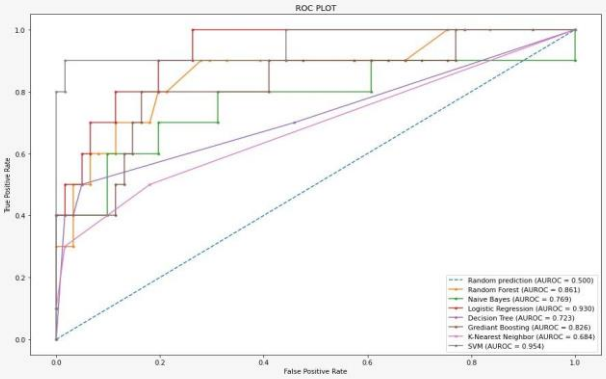


Figure 3. AUC-ROC graph for Ivy Dataset

TABLE IVI

(C) DATASET JEDIT V/S CLASSIFIERS

Datasets	**Jedi**	**Jedit**	**Jedit**	**Jedit**

	t-4.0	-4.1	-4.2	-4.3
Classifiers				
Random (chance) Prediction	0.500	0.500	0.500	0.500
Random Forest	0.615	0.809	0.968	0.703
Naïve Bayes	0.531	0.820	0.945	0.562
Logistic Regression	0.570	0.831	0.975	0.611
Decision Tree	0.575	0.675	0.959	0.495
Grediant Boosting	0.590	0.812	0.972	0.783
Support Vector Machine	0.214	0.868	0.994	0.062
K-Nearest Neighbor	0.675	0.815	0.752	0.479

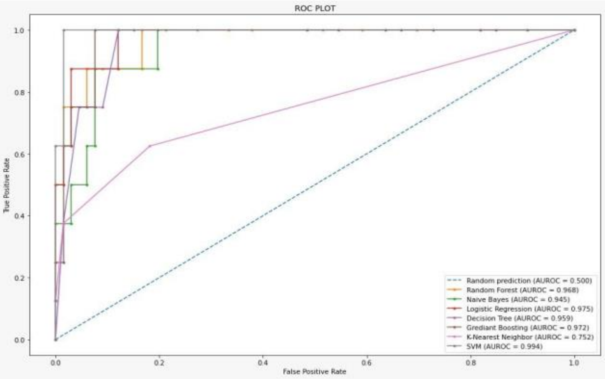


Figure 4. AUC-ROC graph for Jedit Datasets

TABLE VI

(D) DATASET LOG4J v/s CLASSIFIERS

Datasets	**Log4j-1.0**	**Log4j-1.1**	**Log4j-1.2**
Classifiers			
Random (chance) Prediction	0.500	0.500	0.500
Random Forest	0.821	0.804	1.000
Naïve Bayes	0.825	0.741	1.000
Logistic Regression	0.833	0.857	1.000
Decision Tree	0.667	0.750	1.000
Grediant Boosting	0.746	0.812	1.000
Support Vector Machine	1.000	0.920	1.000
K-Nearest Neighbor	0.774	0.741	0.949

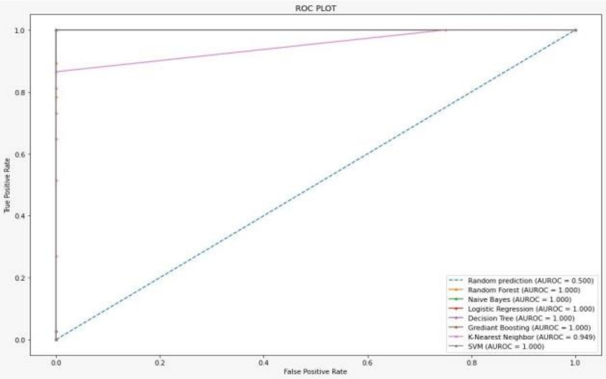


Figure 5. AUC-ROC graph for Log4J Datasets

TABLE VII

(E) DATASET LUCENE v/s CLASSIFIERS

Datasets	**Lucene-2.4**
Classifiers	
Random (chance) Prediction	0.500
Random Forest	0.837
Naïve Bayes	0.752
Logistic Regression	0.785
Decision Tree	0.687
Grediant Boosting	0.781
Support Vector Machine	0.907
K-Nearest Neighbor	0.735

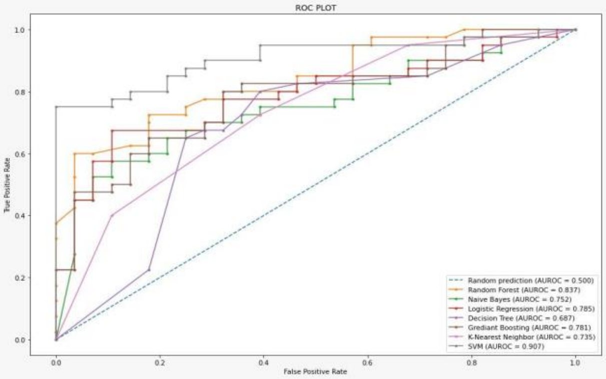


Figure 6. AUC-ROC graph for Lucene Datasets

TABLE VIII

(F) DATASET POI V/S CLASSIFIERS

Datasets	Poi-2.0	Poi-2.5	Poi-3.0
Classifiers			
Random (chance) Prediction	0.500	0.500	0.500
Random Forest	0.663	0.868	1.000
Naïve Bayes	0.720	0.705	0.983
Logistic Regression	0.711	0.787	1.000
Decision Tree	0.543	0.789	1.000
Grediant Boosting	0.616	0.857	1.000
Support Vector Machine	0.230	0.903	1.000
K-Nearest Neighbor	0.626	0.849	0.856

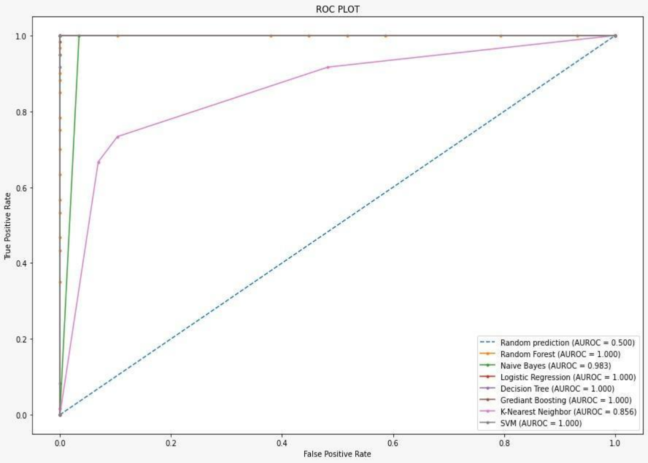


Figure 7. AUC-ROC graph for Poi Datasets

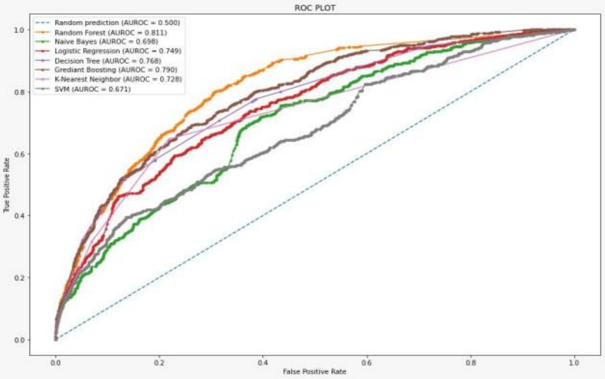


TABLE VIIII

(G) DATASET XERCES V/S CLASSIFIERS

Datasets	Xerces-1.2_1	Xerces -1.3	Xerces -1.4
Classifiers			
Random (chance)	0.500	0.500	0.500
Random Forest	1.000	1.000	0.904
Naïve Bayes	0.986	1.000	0.819
Logistic Regression	1.000	1.000	0.840
Decision Tree	1.000	1.000	0.830
Gradient Boosting	1.000	1.000	0.908
Support Vector	1.000	1.000	0.881
K-Nearest Neighbor	0.956	0.800	0.812

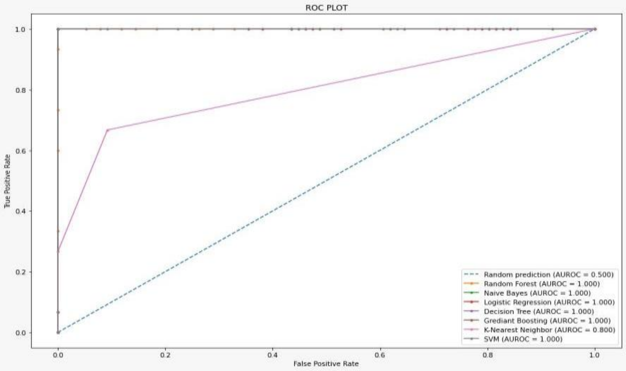


Figure 8. AUC-ROC graph for Xerces Datasets

TABLE IXI

(H) DATASET XALAN V/S CLASSIFIERS

Datasets	**Xalan-2.4-1**	**Xalan-2.5_1**	**Xalan-2.6_1**
Classifiers			
Random (chance)	0.500	0.500	0.500
Random Forest	0.850	0.796	0.856
Naïve Bayes	0.777	0.641	0.798
Logistic Regression	0.769	0.632	0.806
Decision Tree	0.814	0.699	0.787
Grediant Boosting	0.836	0.740	0.842
Support Vector Machine	0.839	0.301	0.873
K-Nearest	0.652	0.687	0.761

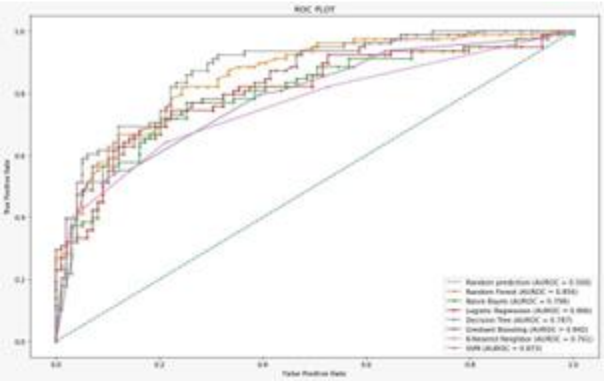


Figure 9. AUC-ROC graph for Xalan Datasets

TABLE XI

(I) DATASET PROP V/S CLASSIFIERS

Datasets	**Prop -1**	**Prop -2**	**Prop -3**	**Prop -4**	**Prop -5**	**Prop -6**
Classifiers						
Random (chance) Prediction	0.500	0.500	0.500	0.500	0.500	0.500
Random Forest	0.811	0.847	0.696	0.789	0.733	0.705
Naïve Bayes	0.698	0.686	0.698	0.692	0.717	0.685
Logistic Regression	0.749	0.719	0.700	0.745	0.712	0.675
Decision Tree	0.768	0.760	0.698	0.735	0.730	0.530
Gradient Boosting	0.790	0.789	0.762	0.783	0.767	0.696
Support Vector Machine	0.671	0.652	0.650	0.598	0.667	0.804
K-Nearest Neighbor	0.728	0.721	0.612	0.670	0.656	0.716

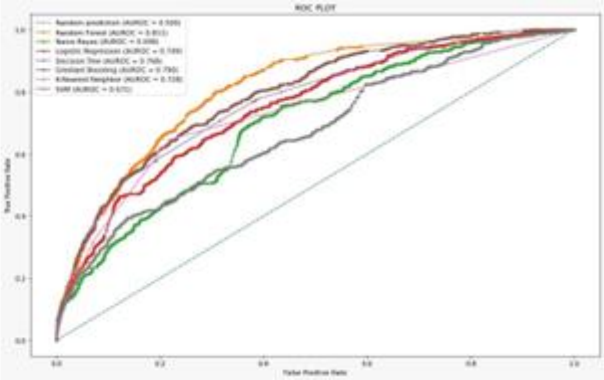


Figure 10. AUC-ROC graph for Prop Datasets

To evaluate model performance researcher used following parameters for model evaluations.

1. Sensitivity / True Positive Rate / Recall (TPR) are termed as correctly classified Negative classes.
2. False Positive Rate (FPR) values are correctly classified positive classes = False Negative Rate.

FPR tells us what proportion of the negative class got incorrectly classified by the classifier.

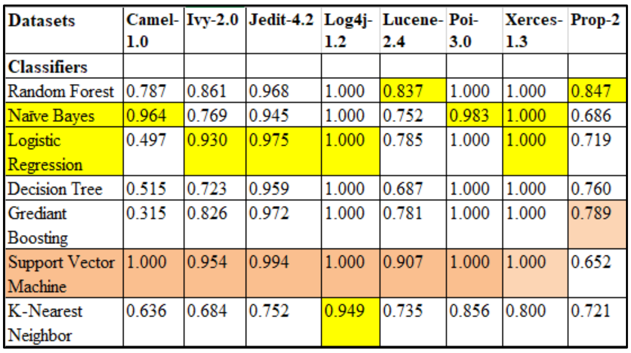
Plotting AUC ROC Curve

After calculating all the parameters, AUC-ROC curve were plotted for all models applied, and comparative study is done.

TABLE XIII
DATASET USED FOR STUDY WITH ALL SPECIFICATION

Dataset for Software with Software Fault							
SN	Dataset name	Ver	KLOC	Total number of Modules	Total Number Modules without Defect	Total No of Modules with Defect	Percentage of Defected Modules (%)
1	Ant	1.7	208	746	580	166	22.25
2	Camel	1	33	340	327	13	3.82
3	Camel	1.2	66	609	393	216	35.47
4	Camel	1.4	98	873	728	145	16.61
5	Camel	1.6	113	966	778	188	19.46
6	Ivy	2	87	353	313	40	11.33
7	Jedit	4	144	307	232	75	24.43
8		4.1	153	313	234	79	25.24
9		4.2	170	368	320	48	13.04
10		4.3	202	493	482	11	2.23
11	Lucane	2.4	102	341	138	203	59.53
12	Poi	2	93	315	278	37	11.75
13		2.5	119	386	138	248	64.25
14		3	129	443	162	281	63.43
15	Prop	1	3816	18472	15734	2738	14.82
16		2	3748	23015	20584	2431	10.56
17		3	1604	10275	9095	1180	11.48
18		4	1508	8719	7879	840	9.63
19		5	1081	8517	7218	1299	15.25
20		6	97	661	595	66	9.98
21	Xalan	2.4	225	724	613	111	15.33
22		2.5	304	804	417	387	48.13
23		2.6	411	886	475	411	46.39
24		2.7	428	910	12	898	98.68
25	Xerces	1.2	159	441	370	71	16.10
26		1.3	167	454	385	69	15.20
27		1.4	141	589	152	437	74.19

TABLE XIIV
AUC-ROC SCORE APPLIED WITH CLASSIFIERS ON DATASETS



Datasets	Camel-1.0	Ivy-2.0	Jedit-4.2	Log4j-1.2	Lucene-2.4	Poi-3.0	Xerces-1.3	Prop-2
Classifiers								
Random Forest	0.787	0.861	0.968	1.000	0.837	1.000	1.000	0.847
Naïve Bayes	0.964	0.769	0.945	1.000	0.752	0.983	1.000	0.686
Logistic Regression	0.497	0.930	0.975	1.000	0.785	1.000	1.000	0.719
Decision Tree	0.515	0.723	0.959	1.000	0.687	1.000	1.000	0.760
Grediant Boosting	0.315	0.826	0.972	1.000	0.781	1.000	1.000	0.789
Support Vector Machine	1.000	0.954	0.994	1.000	0.907	1.000	1.000	0.652
K-Nearest Neighbor	0.636	0.684	0.752	0.949	0.735	0.856	0.800	0.721

Table IV shows that support vector machine (SVM) is more effective, which gives score 1 or approximately 1 followed by Gradient Boosting and Logistic Regression as second highest score. The following table shown contains only 8 datasets which had given us positive results.

VI. CONCLUSION

The software quality improvement is an ongoing process. Research on cross project fault prediction is not done much. As part of this study, a unique approach has been taken into account, where PROMISE depository is used. Total nine algorithms are applied on 27 datasets. Earlier only 4-5 maximum datasets were used to predict the software faults. Also maximum common features were extracted from all datasets to maintain consistency before applying the proposed method.

It is observed that Support Vector Machine algorithms works best on cross projects and gives maximum results. AUC-ROC score of 0.97 to 1 is observed using SVM. The second largest values were observed using Logistics Regression and Naïve Bayes.

Two dimensional area under the ROC curve is measured by AUC. It has the ability to differentiate between faulty and non-faulty classes. Binary classification problem uses AUC metric for model evaluation. The classification models used for experiment, performs better, if AUC curve is having values near 1. Other models having values near to 0, have poor separability quality of buggy or faulty modules. The graph is plotted using AUC-ROC values for all models and datasets for comparative analysis. (Refer table 3).

This paper focused on experimental evaluation of eight fault prediction models which was not done earlier. The score of the given model is also recorded, over the 27 datasets using AUC-ROC. Researcher can compare the highest value and the lowest value from the models applied to the datasets individually. Thus, researcher can decide, the best classification model to be used for accurate faulty predictions in order to classify faulty modules in a software. The datasets would also be selected depending on which type of algorithms suits your model. As this selection will improve the results given. Future research should make use of this detailed analysis, and can study further on cross-project based SFP.

VII. REFERENCES

[1]. Lim, S.Y.; Kiah, M.M.; Ang, T.F. Security Issues and Future Challenges of Cloud Service Authentication. Polytech. Hung. 2017, 14, 69–89.

[2]. Borylo, P.; Tornatore, M.; Jaglarz, P.; Shahriar, N.; Cholda, P.; Boutaba, R. Latency and energy-aware provisioning of network slices in cloud networks. Comput. Commun. 2020, 157, 1–19.

[3]. Carmo, M.; Dantas Silva, F.S.; Neto, A.V.; Corujo, D.; Aguiar, R. Network-Cloud Slicing Definitions for Wi-Fi Sharing Systems to Enhance 5G Ultra-Dense Network Capabilities. Wirel. Commun. Mob. Comput. 2019, 2019, 8015274.

[4]. Dang, L.M.; Piran, M.; Han, D.; Min, K.; Moon, H. A Survey on Internet of Things and Cloud Computing for healthcare. Electronics 2019, 8, 768.

[5]. Srinivasamurthy, S.; Liu, D. Survey on Cloud Computing Security. 2020. Available online: https://www.semanticscholar.org/ (accessed on 19 July 2020).

[6]. Mathkunti, N. Cloud Computing: Security Issues. Int. J. Comput. Commun. Eng. 2014, 3, 259–263.

[7]. Stefan, H.; Liakat, M. Cloud Computing Security Threats And Solutions. J. Cloud Comput. 2015, 4, 1.

[8]. Fauzi, C.; Azila, A.; Noraziah, A.; Tutut, H.; Noriyani, Z. On Cloud Computing Security Issues. Intell. Inf. Database Syst. Lect. Notes Comput. Sci. 2012, 7197, 560–569.

[9]. Palumbo, F.; Aceto, G.; Botta, A.; Ciuonzo, D.; Persico, V.; Pescapé, A. Characterizing Cloud-to-user Latency as perceived by AWS and Azure Users spread over the Globe. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Taipei, Taiwan, 7–11 December 2019; pp. 1–6.

[10]. Hussein, N.H.; Khalid, A. A survey of Cloud Computing Security challenges and solutions. Int. J. Comput. Sci. Inf. Secur. 2017, 1, 52–56.

[11]. Le Duc, T.; Leiva, R.G.; Casari, P.; Östberg, P.O. Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey. ACM Comput. Surv. 2019, 52, 1–39.

[12]. Li, K.; Gibson, C.; Ho, D.; Zhou, Q.; Kim, J.; Buhisi, O.; Gerber, M. Assessment of machine learning algorithms in cloud computing frameworks. In Proceedings of the IEEE Systems and Information Engineering Design Symposium, Charlottesville, VA, USA, 26 April 2013; pp. 98–103.

[13]. Callara, M.; Wira, P. User Behavior Analysis with Machine Learning Techniques in Cloud Computing Architectures. In Proceedings of the 2018 International Conference on Applied Smart Systems, Médéa, Algeria, 24–25 November 2018; pp. 1–6.

[14]. M. T. Khorshed, A. B. M. Shawkat Ali, and S. A. Wasimi, “Trust issues that create threats for cyber attacks in cloud computing,” in Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS, 2011, doi: 10.1109/ICPADS.2011.156.

[15]. C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, “A survey on security issues and solutions at different layers of Cloud computing,” J. Supercomput., vol. 63, pp. 561–592, 2013, doi: 10.1007/s11227-012-0831-5.

[16]. M. T. Khorshed, A. B. M. S. Ali, and S. A. Wasimi, “A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing,” Futur. Gener. Comput. Syst., vol. 28, no. 6, pp. 833–851, 2012, doi: 10.1016/j.future.2012.01.006.

Cite this article as :

Dr. M. Aruna Safali, "An Empirical Study of Classification Models Using AUC-ROC Curve for Software Fault Predictions", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 9, Issue 1, pp.261-271, January-February-2023.