Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

A Practitioner Approach of Deep Learning Based Software Defect Predictor

Yashwant Kumar [a], Dr. Vinay Singh [b]

a Department of computing and Information Technology, Usha Martin University, Ranchi, Jharkhand
b Associate Professor, Department of computing and Information Technology, Usha Martin University, Ranchi, Jharkhand

ABSTRACT In Software Development Life Cycle (SDLC), the coding plays the very crucial phase so far as quality of software is concerned. The quality of software highly depends on the quality of coding done by the software developer. Minor defects in software may results in huge loss to software development firm. To test phase of the software development life cycle is very much required, it is a mechanism of quality control system in SDLC. Early detection of defect in software, mostly during development saves the time of testing and increase the development efficiently. There are lots of ML Model developed by researchers for said purpose. Natural Language Processing (NLP) based Software Defect Predictor outperformed the Traditional ML based techniques. Also Deep Learning based feature extractor has outperformed the hand crafted Software metric.

INDEX TERMS : Machine Learning, Software Defect Prediction, Natural Language Processing, Software Metrics.

I. Introduction

Software Defect Prediction (SDP) has become very vital activities in Software Development Life Cycle (SDLC). The Testing of a large scale software product requires lot of resources and it involvestime consuming activities, that may results in delayed or failure in product launch. Early detection of fault in software during development phase helps the teamto minimizes the cost of testing and improves the effectiveness of software development process.

In Most of Research work,Machine Learning (ML) Techniques[1]arewidely used for Software Defect Prediction. As Software Defect Predictor is a Supervised Learning Techniques in context of Machine Learning so it requires lots of historical data to train the good models. Historical Data of Software Defect is actually labeled data set of PreviousSoftware Project with either Defective or Non-Defective information, so Software Defect Prediction is aclassification Problemin parlance of ML Techniques. Many Researcher's has also tried to present it as regression problem in terms of number of errors found in the modules,[2] but by and large it is

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

presented as binary classification problem which can classifythe module as either Defective(D) or Non-Defective(N).This can save lot of time and energy for the Testing Team.

The defect dataset is generated from Source Code of the Software. The data consists of featured data and labeled data (Buggy or Clean) . These feature data is used to determine whether software is defective or not. There are two ways we can extract feature data. Manually extracted features and Deep-learning-generated features. The manually extracted feature are traditionally calculated from Software Metrics. Like MOOD, CK, Halstead, McCabe etc.[3]Whereas DEEP-learning methods are based on Natural Language Processing (NLP) based pre-processing techniques. Like One-Hot-Encoding, Word-to-Vector, Glove Encoding and Various Embedding Techniques like Skip Gram or Continuous Bag of Words (CBOW).

These extracted featured datasets is fed into either Neural Network or Traditional Machine Learning Based Techniques. The Traditional ML based Techniques mostly used for such type of classification modelsare Support Vector Machine (SVM), Naive Bayes Classifier (NB), Decision Tree Classifier (DTC), Logistic Regression and Ensemble Techniques Likes Random Forest, Bagging and Boosting.

Neural Network Based Techniques are mostly based on Deep Learning Methods which Includes the Set of Input Layer and Multiple Hidden Layers and Classification based Output Layer. In recent years Recurrent Neural Network (RNN) based NLP techniques like Long-Short-Term-Memory (LSTM), LSTM with Attention Layer is widely explored to solve SDP related problem with high accuracy.[4]

LSTM based model is sequential learning methods and widely used in NLP for Text Generation and Natural Language Translation task.

The SDP model should predict the defect in Within Project and Cross Projects Modules. Within Project Defect Prediction(WPDP) is relatively efficient due to availability of historical defect dataset of previous versions.

To prepare the data from Source code of software, to input in NLP models, the most widely accepted methods used by researchers are Abstract Syntax Tree (AST) of Programming language.[3]The AST is programming language neutral and very useful for Cross Project Defect Predictions, where main issue is either non-availability of historical data or very few historical data. [5]

The main challenge in training of SDP model is availability of historical data in new software product. Whichcan be solved by AST based representation of source code of another software product, that can help to transfer the learning model to new project where no versionsareavailable as historical data.

The remainder of the paper explained each of these concepts in details.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

II. Software Defect Prediction

The Software Defect Prediction is a set of techniques, which expose the probability of mistakes in Software System using Machine Learning Methods. In other words it is used to establish the relationship between software metrics and bugs. [6]

Several large software companies that won't review software modules unless the SDP models predict that there are high percentage of fault prone. Hence, defect detectors have a major economic impact when they may force programmers to rewrite the codes.

The defect dataset prepared for defect prediction consists of columns for software metrics (basically used for defect prediction) and one column having value either 0 (Defect) or 1 (Non-Defect).

The embedding of source code is another method for extracting the pattern/features in the source code along withone column, that keeps the probability of defect or non-defect labels, which is further used for training the SDP Models. [7]

A. Software Metrics for Defect Prediction

The Software metrics (i.e. features) are hand crafted and manually designed to measure the software entity. It is a quantitative measurement that assigns numbers or symbols to attributes of the measured entity.This entity can be source code of application or a software development process activity. Many previous researchers have pointed out that there is a relationship between software metrics and defect predictions.Software metrics can be classified as static code metrics and process metrics.

There are variousstatic code metrics derived from source code, andbeing introduced by researchers, mostly used for SDP. Some of the potentialsource code metrics are listed below.[8]

Introducer	**Metric Name**	**Descriptions**
M.H. Halstead : base measures	mu1	number of unique operators
	mu2	number of unique operands
	N1	total occurrences of operators
	N2	total occurrences of operands
	length = N	N1 + N2
	vocabulary = mu	mu1 + mu2
	mu1'	potential operator count (just the functionname and the"return" operator)
	mu2'	potential operand count. (the number of arguments to the module)
M.H. Halstead :Derived	volume = V = N * log2(mu)	the number of mental

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

Introducer	Metric Name	Descriptions
measures:		comparisons needed to write a program of length N
	V* = (2 + mu2')*log2(2 + mu2')	Volume on minimal implementation
	L = V*/N	program length
	D = 1/L	difficulty
	L' = 1/D	Inverse difficulty
	I = L'*V'	intelligence
	E = V/L	effort to write program
	T = E/18 seconds	time to write program, time estimator
M.H. Halstead : lines of code measures	LOCode	line count
	LOComment	count of lines of comments
	LOBlank	count of blank lines
	LOCodeAndComment	line count + count of lines of comments
	branchCount	Number of the flow graph
McCabe Metric	v(G)	Cyclomatic Complexity
	ev(G)	Essential Complexity
	iv(G)	Design Complexity
	loc	Linecount of code
CK metric	WMC	Weighted Method per Class
	DIT	Depth of Inheritance Tree
	NOC	Number of children
	CBO	Coupling between objects
	RFC	Response for a Class
	LCOM	Lack of Cohesion in Methods
MOOD Metric	AHF	Attribute Hiding Factor
	MHF	Method Hiding Factor
	MIF	Method Inheritance Factor
	AIF	Attribute Inheritance Factor
	COF	Coupling Factor
	POF	Polymorphism Factor

B. Process Metrics for Defect Prediction

Process metrics can be extracted from Source Code Management system(like Git) based on historic changes on source code overtime. Source code management System (SCMS) is used to maintain trails of modifications to a source code repository. SCMS keeps the history of changes to a code base and helps resolve conflicts when merging updates from multiple developers. SCMS is also called as Version control System (VCS). Some researcher suggested that the Change (process) metrics are more efficient than static code metrics for defect prediction

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

because process data contains more discriminatory information about defect distribution than the source code itself.

The Some of the popular process metrics used for SDP are listed below[9]

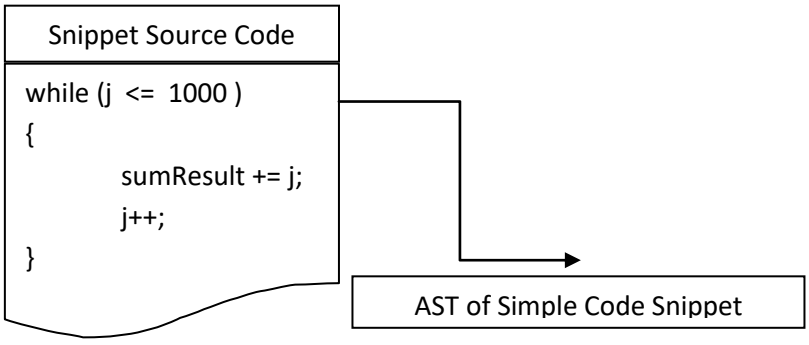
Category	Metric Name	Descriptions
Author-Ship	OWN	Owner's Contributions
	EXP	Developer's Experience
	DDEV	Number of Distinct Developers
	ADEV	Number of Active Developers
Change Type	NREFAC	Number of Refactoring Changes
	NBF	Number of Bug Fixing Changes
Change Interval	MAXI/MINI/AVGI	Max/Min/AVG Time Gap between two changes
Code Churn	ADD	Lines of code added
	DEL	Lines of code deleted
	HCM	Entropy of Multiple Changes
Co-Change	MCO	Maximum Number of Files co-changed
	ACO	Average Number of Files co-changed
	NDDEV	Co-Change in properties of files like ownership
Semantic Change Type	COND	Number of condition expression change
	ELSE	Number of ELSE part changes.

C. Software Feature Extraction for Defect Prediction

Feature Extraction is a DEEP Learning based model used to generate featured data for Software Defect Prediction. It is a powerful technique to extract semantic and syntax features hidden in source code. The Deep learning based methods automatically encode the feature from source code (Software Feature) or from change sequence (Process Data) from Source Code Management System.

To extract the semantic and syntax feature of source code, the most popular methods used by researchers is Abstract Syntax Tree (AST) generated from source code.

III. Overview of Process, AST Conversion from Source Code to Feature Data



Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

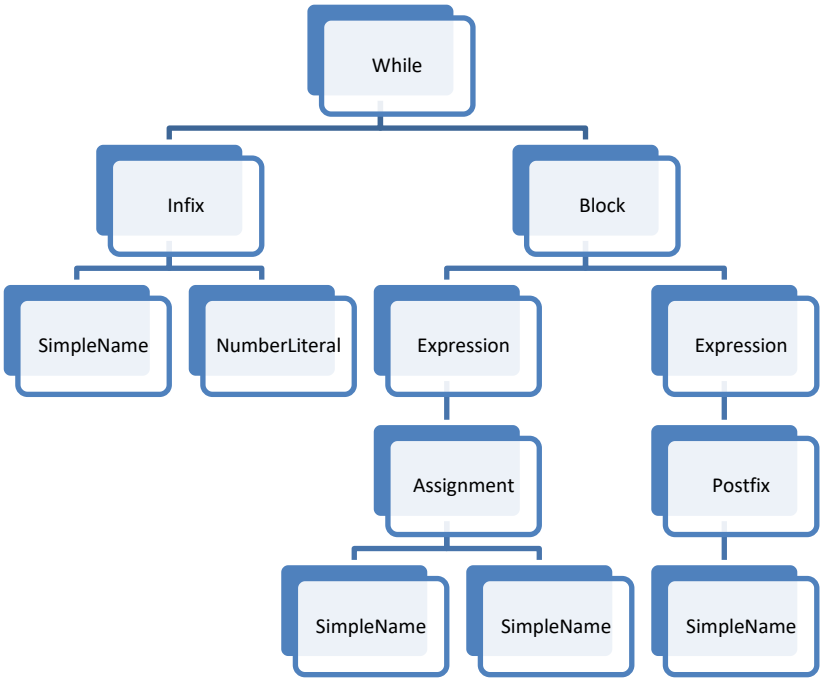


Figure: Source Code Conversion to AST

Vocabulary (While, Infix, Block, SimpleName, NumberLiteral, Postfix, Expression, Assignment)

Mapping Table

Mapping Table Can be Created Using TF-IDF, CBOW, and most effective Deep Learning based Methods Like Word2Vecas Follows

Vocabulary	**Vector Representations (Size of the Vector is Number of Output in Output Layer)**
While	[-0.02878454 -0.01797051 0.00237926 -0.00371939 0.00606985 -0.04638186 0.03167989 -0.04400144 -0.04536068 0.0120385]
Infix	[0.03788391 -0.0389317 0.02806017 0.00850868 -0.00974257 -0.01735647 -0.02383494 0.01912074 0.04257665 -0.01970553]
Block	[-0.04847089 -0.04180741 0.03103842 -0.04266282 0.00140287 0.03663715 -0.00899701 0.02680084 0.03196992 -0.03501695]
SimpleName	[-0.00751654 0.02754606 -0.05469051 -0.00526549 0.01212245 -0.0509387 0.02762272 0.00043802 0.03384242 -0.04551497]
....	....

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

TrainingDataset

Features	Label
V1,V2,V3.....Vn	1
V1,V2,V3.....Vn	0
V1,V2,V3.....Vn	0
V1,V2,V3.....Vn	0
V1,V2,V3.....Vn	1
.....	...

Test Sample

Features
V1,V2,V3.....Vn

The above process depict how source code is converted into Vectors, that can be passed as Input to Various ML Algorithms. The meta data of source project is extracted from Version Control System (VCS) and Bug Tracking System (BTS) (e.g. Jira, Bugzila).(D. Chen et al., 2019) The VCS keeps track of all the source code of the modules and BTS keeps the information about the defect labels of concerned module. This forms the data set to be used for Pre-Processing. The dataset of source code so obtained is converted into Token Vectors and attached with defect labels. This forms the Pre-processed Data for the Defect Prediction Models. The Pre-Processed data is split into Training Set and Test Set. The Training Set is fed into ML Models to train and Test Set is used to Validate the Model.

IV. Evaluation Metric

The performance of Classification based Supervised SDP models is measured by various evaluation metric. The most frequently preferred evaluation metrics by researchers are Recall, Precision, F1-score and AUC.[10] . These evaluation metrics is based on confusion matrix. For binary classification of SDP, if Defect is represented as 0 (Positive) and Clean or Non-Defect is represented as 1 (Negative),then there are total four types of output possible.

1. True Positive : Model predicts the instance as Defective (0) and it is actually Defective. (0)
2. True Negative: Model predicts the instance as Non-Defective (1) and it is actually a Non-Defective (1)
3. False Positive : Model predicts the instance as Defective (0) but actually it is Non-Defective (1)
4. False Negative: Model predicts the instance as Non-Defective (1) but actually it is Defective. (0)

	Predicted Class

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

		0	1
Actual class	0	True Positive	False Negative
	1	False Positive	True Negative

Figure: Confusion Matrix

Recall refers to the Ratio of number of cases, which are correctly classified as buggy.

$$recall = \frac{truepositive}{truepositive + falsenegative}$$

Precision refers to the ratio of number of cases classified to be buggy.

$$precision = \frac{truepositive}{truepositive + falsepositive}$$

F1-Score is the weighted harmonic average of them.

$$F1Score = \frac{2\ xRecallxPrecision}{Recall\ +\ Precision}$$

AUC is the trade-off between true positive rate (TPR) or recall, and false positive rate (FPR).

$$FPR = \frac{FalsePoitive}{FalsePositive + TrueNegative}$$

AUC is very powerful to represent the class distribution and reduce misclassification costs. It is mode widely used Evaluation metric for Defect Prediction.AUC close to 1 is considered has best model for classification.

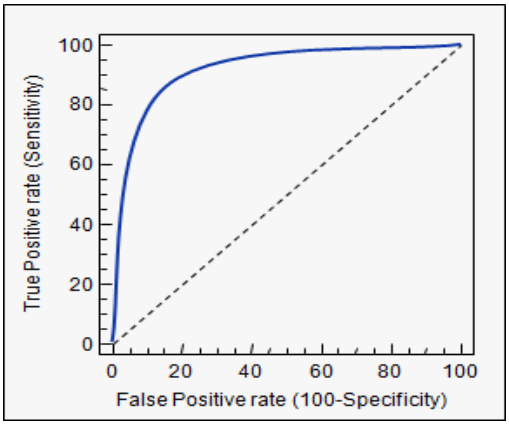


Figure : Area Under the Curve (AUC)

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

V.Comparison of Traditional Software Defect Predictor

Authors	Data source	Metrics	Algorithm/Method	Result
[11]	Six publicly available software defect datasets: 1. Ant-1.7. 2. Camel-1.6. 3. KC3 datasets. 4. MC1. 5. PC2. 6. PC4.	These Dataset had total 51 different Metrics (LOC, McCabe's CC, Halstead difficult, Condition Count, Branch count, Number of unique operands) and OO Metrics (C.K. Metrics and MOOD Metrics) (Weighted methods for class, Depth of inheritance tree, Inheritance coupling, Number of children)	The Ensemble system incorporates 7 classifiers: Random forests (RF), Gradient boosting (GB), Stochastic Gradient Descent (SGD), weighted SVMs (W-SVMs), Logistic regression (LR), Multinomial Naive Bayes (MNB) and Bernoulli Naive Bayes (BNB) as base classifier. It uses Greedy forward selection (GFS) as a feature selection technique.	The highest AUC results was attained by the proposed model against the PC2 dataset of AUC measure of 0.91.
[12]	There were two dataset used one from the MIS dataset and the KC2 dataset.	They used 21 software metrics of the KC2 dataset (e.g., LOC, V(G), EV(G), etc.), The MIS dataset contains 12 metrics.	This paper proposed fully connected neural network To predict the number of defects in a software module.	The mean squared error (MSE) of MIS varies from 66.30 to 46.01, and the R^2 of MIS varies from 0.32 to 0.42. Similarly, the MSE of KC2 varies from 0.13 to 0.109, and the R^2 of KC2 varies from 0.193 to 0.297.
[2]	It used NASA datasets. They are MC2,PC1, KC1, PC3,MC1,PC2.	Logical line count, Cyclomatic complexity, Halstead difficulty and Halstead length are mostly preferred in this paper as compared to total number of lines.	It introduced the methods of Cost Sensitive Voting (CSVoting) and Cost Sensitive Forest (CSForest).	It is shown that CSForest coupled with CSVoting produced the lower cost predictions than the existing techniques.
[13]	Four NASA Datasets, Two datasets (PC1 and JM1)are from software projects written in a	metrics are McCabe, McCabe and Butler, Halstead metrics, Total 21 metrics has been used.	This paper presents the application of hybrid artificial neural network (ANN) and Quantum	The AUC value distributions on for (QPSO + ANN) model for PC1,JM1,KC1,KC3 dataset are 0.899,0.777,0.791 and 0.862

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

	procedural language (C) The other two datasets(KC1 and KC3) are from projects written in object-oriented languages (C++ and Java)		Particle Swarm Optimization (QPSO) in software fault-proneness prediction. QPSO is applied for reducing dimensionality.	respectively. The time complexity value of Model (QPSO + ANN) for PC1,JM1,KC1,KC3 is .29, 6.13,2.85,1.46 respectively
[1]	10 open-source projects with 34 releases dataset, available at the PROMISE repository are used. Namely Ant,Camel, Ivy,Jedit,Lucene, Poi,Synapse,Velocity,Xal a and ,Xerces Projects	20 static code metrics, they are CK suite (WMC,DIT,LCOM,RFC,CBO,NO C), Martin's metrics (CA,CE), QMOOM suite (DAM,NPM,MFA,CAM,MOA), Extended CK suite (IC,CBM,AMC,LCOM3), and McCabe's CC (MAX_CC,AVG_CC) as well as LOC.	The simple static code metric such as LOC has been validated to be a useful predictor of software defects, it defines progressive reduction on the size of feature set as metric set simplification.	Minimum metric subset is ideal because of its ability to provide good results in different scenarios and being independent of classifiers. Their results shows that simple classifiers such as Naïve Bayes are more suitable to be the choice for defect prediction
[14]	The 15 datasets of CC are (ant, arc, camel, elearn, jedit, log4j, lucene, poi, prop6, redact-or, synapse, system, tomcat, xalan, xerces).	Static code attributes are based mainly on object-oriented metrics including weighted methods per class, number of children, lines of code, etc.	Double Transferring Boosting (DTB) algorithm, which is the extension of AdaBoosting Sequential Ensemble Technique. imbalance data oversampling (SMOTE) is used	The Proposed model achieved PD,PF, G-Measure and MCC as 0.702,0.330, 0.664 and 0.282 respectively.
[15]	The Apache data set the independent variables are nominal type and dependent variable is numerical type.	No specific metrics were mentioned in the paper.	Grouping the different range of performance value into four(1-4) groups. Four commonly used machine learning techniques were compared using WEKA	The prediction accuracy obtained for untrained data set in Neural Network is high.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

			tool. J48, Simple Cart, Multilayer Back-propagation NN and Naive Bayes. Neural network was created using MATLAB.	
[16]	They focused on actual software project data sets	No Specific Metrics is mentioned in this paper.	They have compared the methods of reliability assessment based on neural network with that of deep learning.	They have shown that the proposed method based on the deep learning can assess better than that based on neural network.
[17]	The details of 9 open source projects used were (Caffeine, Fast-Adapter, Fresco, Frezzer,Glide, Design-Patterns, Jedis, Mem-Cached Client, MPAndroidChart) taken from Git.	The study uses a set of 14 Object-Oriented, Inheritance and other metrics to develop defect prediction model.	The Experiments was setup to compare performance of prediction models developed using 14 machine learning techniques (Perceptron, Widrowhoff, back propagation, LVQ1, multipass LVQ, hierarchical LVQ, SOM, multipass SOM, AIRS 1, AIRS 2, CLONALG, CSCA, Immunes1, Immunes2, Immunes99).	The AUC values of Single layer perceptron were between .0852-.0997. It shows that Single layer perceptron outperformed over other ML techniques.
[18]	In this study, they used thirteen datasets including JM1, PC4, KC2, MC1, KC1, PC3, CM1, MW1, PC1, Class, MC2, KC3 and PC2 from NASA.	Some common OO metrics used were :- WMC,NOC, DIT, CBO, RFC, SLOC or LOC, LCOM etc.	Choosing the machine learning techniques: Logistic Regression, K-nearest Neighbors, Decision Tree, Random Forest, Naïve Bayes, Support Vector Machine	The results could realize that SVM achieved the best F1 value. Multilayer Perceptron was the best technique in predicting errors for method-level datasets compared to other techniques. The ROC curve for Multilayer Perception gives the

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

			and Multilayer Perceptron;	highest AUC value (0.91).
[19]	This paper states that no complete and integrated dataset for showing all of the data metrics.	In the static platform, features of code structure are measured as metrics. Static measurements are a number of supervisors and a number of bunches . Dynamic platforms measure testing perfectionism. Basic element measurements depend on auxiliary and information stream scope . The connection between product measurements and blame inclination.	Hybrid of the Machine learning-based faults prediction model using the MLP and PSO algorithms in IoT applications.	Experimental results showed that the proposed verification method has minimum verification time and memory usage for evaluating critical specification rules than other research studies.
[20]	The model is tested on 4 publicly available datasets from the PROMISE repository. There were cm1, kc1, mc2, pc1.	This prediction is done using different software metrics. The commonly used software metrics are McCabe metrics, Halstead metrics and CK metrics.	Investigated the effect of resampling technique on different datasets. Five classifiers namely logistic regression, K Nearest Neighbor (KNN), Decision tree, Multinomial naive bayes (MNB) and Naive Bayes (NB) used.	It is Shown that the model averaging method has performed much better in terms of performance measure as compared to stacking and voting.
[21]	NASA data sets (JM1, KC3, MC1).	There are mainly process-oriented McCabe, Halstead, and object-oriented CK (Chidambaram Kemmerer) metrics were used in datasets.	Compares the performance indicators of LWL, C4.5, Random forest, Bagging, Bayesian Belief Network, Multilayer Feed forward Neural Network, and SVM algorithms.	Study of supervised learning software prediction algorithms, methods of solving imbalanced classification are analyzed.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

VI. Emerging Deep Learning Based Software Defect Predictor

A. Natural Language Processing (NLP)

NLP is analysis of natural language using computers, for example

- Machine Translations
- Spell Check (Auto Correct)
- Automated query answering (Chartbot)
- Speech Recognition
- Speech Parsing
- Sentiment Analysis
- Text Generation etc

NLP is based on :

- Probability and Statistics
- Deep Neural Network
- Machine Learning
- Linguistics
- Common Sense

NLP based techniques is being used for solving problems from various domains. Because it is based on Linguistics, so it is also being explored for software defect because it could understand the semantics and syntax of the programming languages. Like the natural language the source code of the project is resembled as corpus, individual file as document, a list of unique node in AST is vocabulary, and AST is the language model i.e how the nodes are supposed to be organized.

Comparison of NLP Task in Natural Language and Programming Language

Source of data	**Natural Language Task**	**Programming Language Task**
Corpus	Extract Documents	Extract Source Code File
Documents	Extract Sentences	Extract Abstract Syntax Tree
Sentence/AST	Extract Tokens	Extract Token
Token	Syntax Tree	Form Vector representation of Statements
Document	Classification: Sentiment Analysis (Positive or Negative) , Topic Extraction	Defect Prediction (Buggy or Clean)

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

Encoding is the technique of moving sparse (e.g. one hot) to dense vectors. Using Deep Network Method sparse vector is converted into dense vector. Word2Vec encoding can find the syntactic and Semantic relationship among words. Large dimension is more expressive but small dimension of vector train faster. Glove (Global Vector) is more accurate than Word2Vec. The Glove for various programming language is not yet available.

VII. Research Questions

To evaluative the effective of NLP, in Defect Prediction, we investigate following two research questions.

RQ1: Why NLP is being used as Research Techniques to solve Software Defect Prediction Problem, which erstwhile used Tradition ML or Other Deep Learning Methods?

RQ2. Why there were a need to insert NLP Based layers to overcome SDP challenges?

Here is a answer to it. Since NLP is useful in processing sequential data, and it is based on recurrence, but we need to understand that why recurrence is necessary in processing source codes. Source code can be represented as sequential one dimensional discrete index. Where each data points can be represented as Vector. The data points a basically the nodes in AST. The number of data points in a series can be variablesbuthave some specific position in a series.

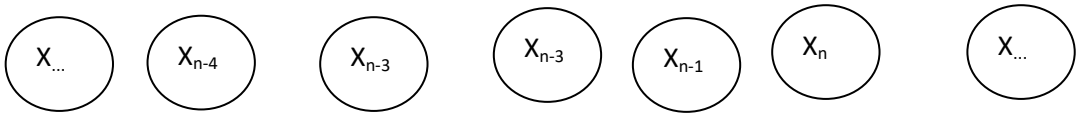


Figure: Sequential Data Points

The some other example of sequential data are Speech, Natural Language Text, Music, Protein and DNA Sequences, time series data like Stock prices etc.

Traditional ML is one-to-one like you have input feature data, then you have some Predictive Algorithm Function of Supervised Learning and Finally the output classified data.

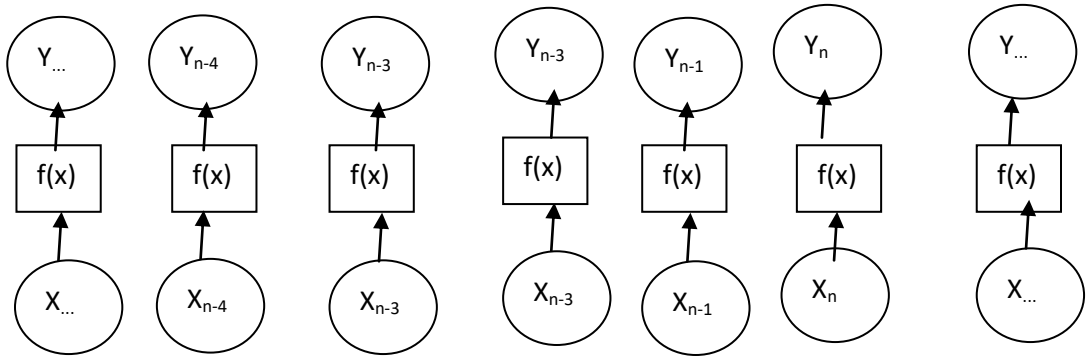


Figure: Traditional ML Representation

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

In traditional ML the past occurrence of input or output data is not taken into account. In source code it is very important to track what came after and what came before like conditional statement before loop statement or after loop will have different results, also position of operator in a expression will have different meaning. Like with hand crafted metric, two codes with different position of operator will have same metric value.

For Example in below two code snippets

```
void fun1 (Stack s)
{
  for(int i=0;i<10;i++)
  {
      s.push(i)
      print(s.pop());
   }
}
```

```
void fun2 (Stack s)
{
  for(int i=0;i<10;i++)
  {
      print(s.pop());
s.push(i)
   }
}
```

The Hand crafted metric like Line of code (loc), operator count, operand count are same in both, but sequences are different, which results in different output.So considering the sequence of code is also important.

Somehow in traditional machine learning for sequence, we may use fix sliding window like below.

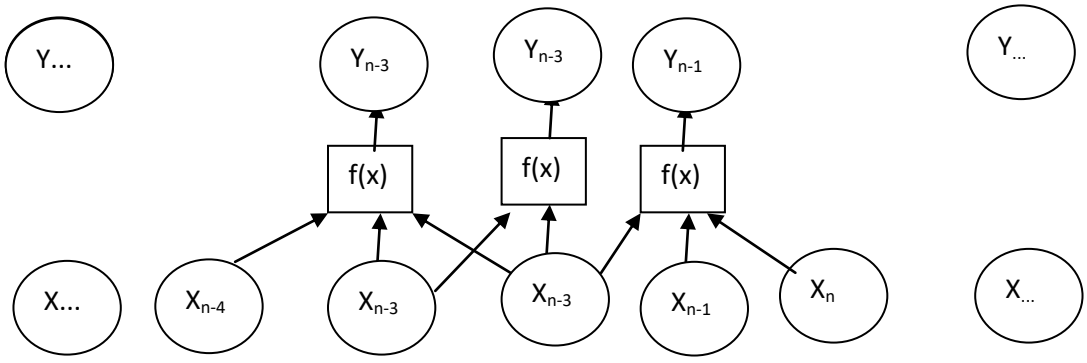


Figure: Traditional ML on Sequential Data

But this model does not take into account the influence of distant past data point. This model is not good for many-to-one model like sentiment analysis and Defect predictions.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

Another method in Traditional ML for sequences is to form a some short of histogram and convert sequence into vector to get fix output of variable length data. But using this method does not consider the influence of order of data because the change in the order does not have any effect on the vector representation.

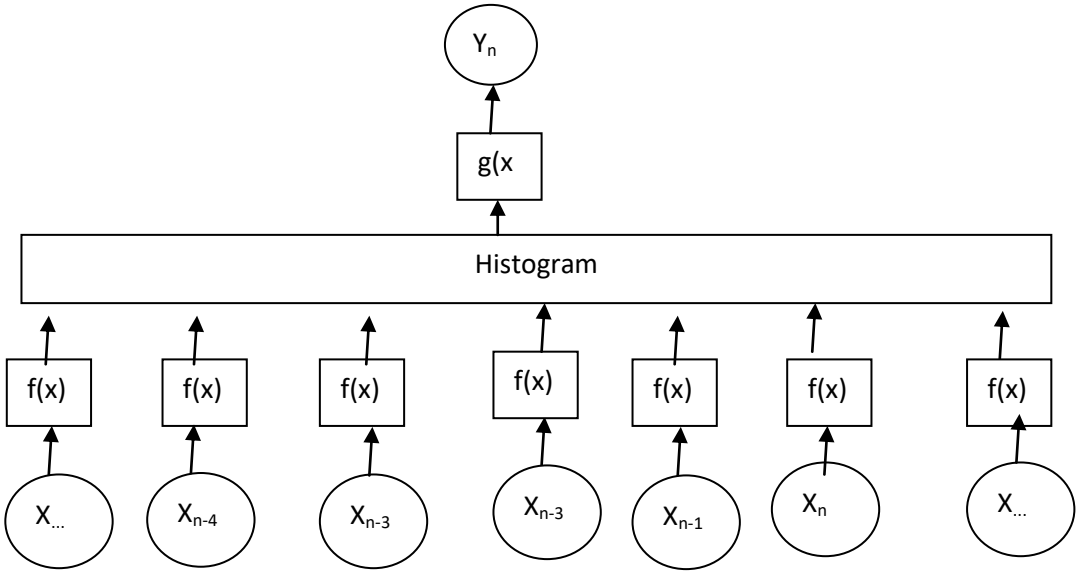


Figure: Using Histogram in Traditional ML on Sequential Data

Introduction of memory (recurrence or state) in neural networks computes the memory state in addition to an output, which is sent to the next time instance. The number is past memory instance is configurable, so even distant data point in the series can be taken into account to get the defective state of the modules. In the most basic form, memory state are simply a hidden neurons (h).

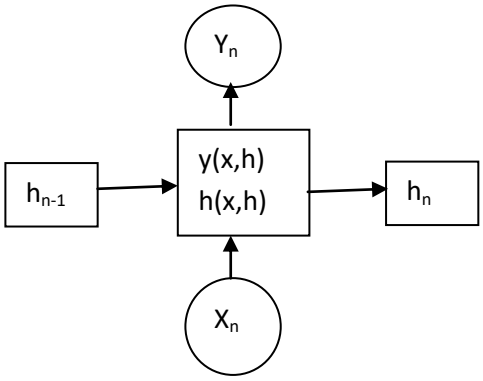


Figure: RNN on Sequential Data (Basic Model)

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

SDP is considered as many-to-one analysis of sequential data using "recurrence". As many sequential statement in a software module will classify the source code as defective or non-defective.

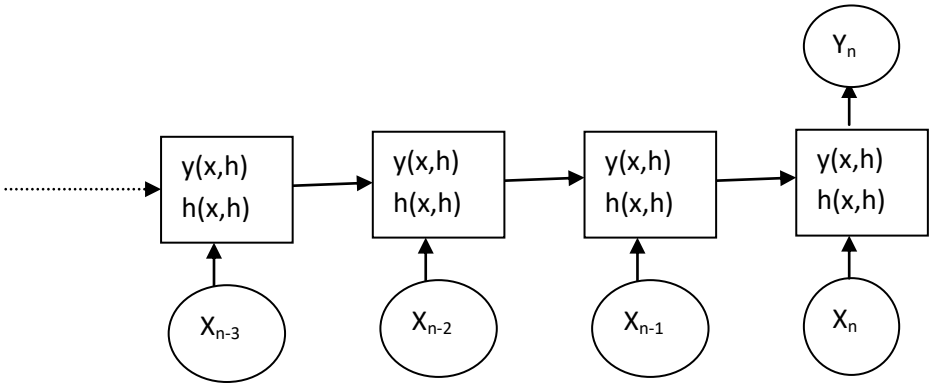


Figure: Many to One, RNN on Sequential Data (Example Model)

Source code of the programming language is a Sequential data, NLP provides the models that is used to process the sequential data. NLP model is designed to have memory to process sequential data. The algorithms or layers of deep learning in NLP includes the Recurrent Neural Network (RNN), LSTM, and Advanced LSTM. Below section describe it in more details.

Recurrent Neural Network (RNN)

Back Propagation through Time (BPTT) is used to tune the weight in RNN. The major drawback of RNN is vanishing or exploding gradient while tuning weight in RNN.

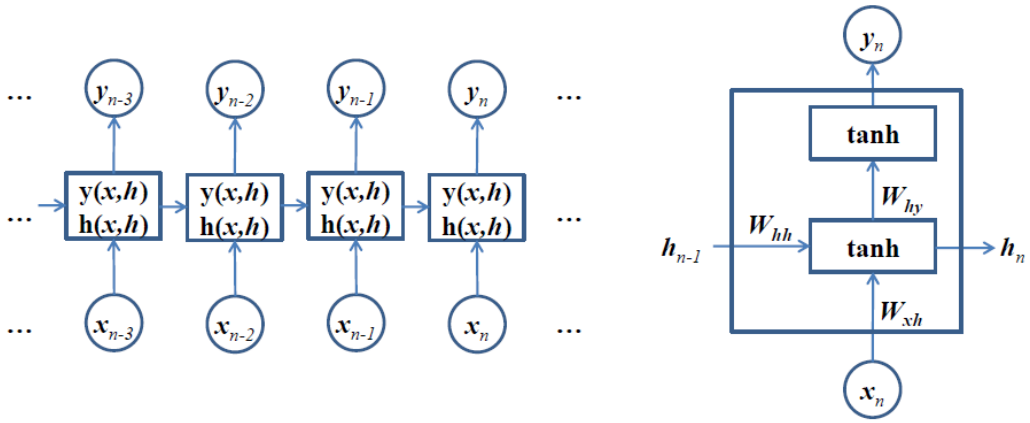


Figure: RNN on Sequential Data (BPTT)

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

LSTM (Long Short Term Memory) in Defect Predictions

LSTN is a special structure of RNN with forget Gates. These gates can be used to avoid the issue of vanishing or exploding gradient found in basic RNN models. A gate is usually sigmoid function which output is either 0 or 1,i,e. *on* or *off*.

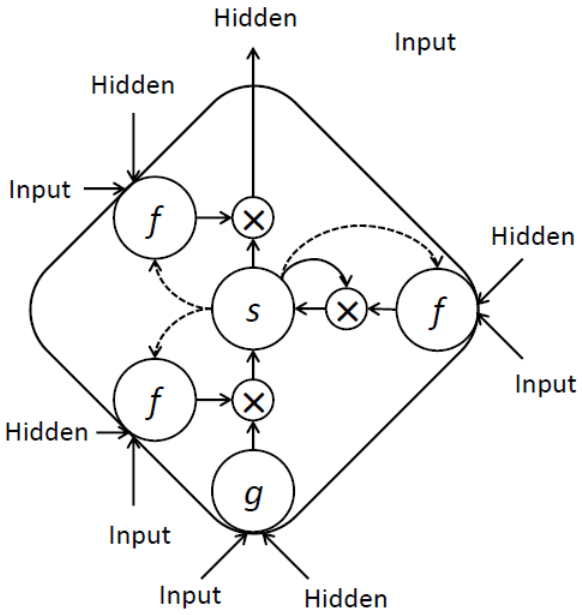


Figure: RNN on with Three Gates Representation

SDP Model Representation in LSTM

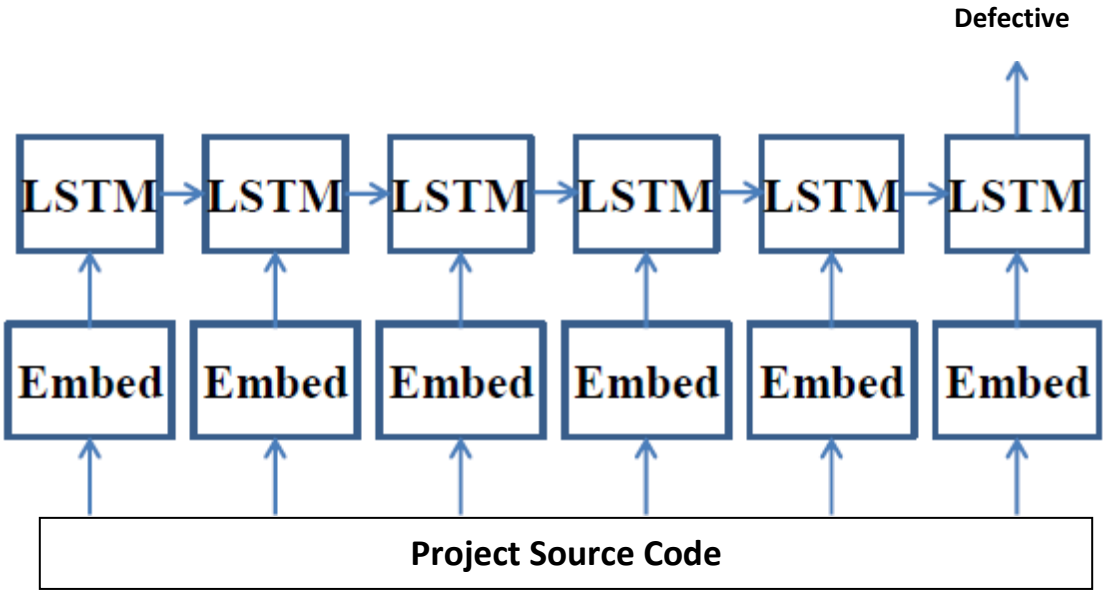


Figure: LSTM Model for SDP

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

There are various advance structure of LSTM, which can be used like Multilayer LSTM where more than one LSTM hidden layers can be used. Bidirectional LSTM which allows reverse flow of information as well, like context from future nodes. LSTM with attention Mechanism, which also takes context of the instance into consideration.

There are some drawbacks also with LSTM like as follows:

- Separate LSTM for separate programming language
- High Training Loss
    - Too few hidden layers
    - Only one hidden layers
- Over fitting
    - Model has too much freedom
        - Too many hidden nodes
        - To many blocks
        - To many layers
        - Not-bi-directional

VIII. CONCLUSION AND FUTURE SCOPE OF WORK

In this article, we tried to present the importance of Deep Learning based methods to predict the software defect during the early stage of software development life cycle. Natural Language Processing has natural Resemblances with Programming Language. So NLP can be used to understand the semantic and syntax feature of programming language and capability to develop the generic software defect predictor. The parsing of Programming language into common Syntax Tree, independent of language specific keywords, can make the models language independent and generic in nature. That can be used for Within Project and Cross Project Defect Prediction by minor modification in Model using Transfer the learning Techniquesby customizing certain language specific layers.

REFERENCES

[1] 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, Mar. 2015, doi: 10.1016/j.infsof.2014.11.006.

[2] M. J. Siers and M. Z. Islam, “Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem,” *Information Systems*, vol. 51, pp. 62–71, Jul. 2015, doi: 10.1016/j.is.2015.02.006.

[3] Z. Cai, L. Lu, and S. Qiu, “An Abstract Syntax Tree Encoding Method for Cross-Project Defect Prediction,” *IEEE Access*, vol. 7, pp. 170844–170853, 2019, doi: 10.1109/ACCESS.2019.2953696.

[4] H. Liang, Y. Yu, L. Jiang, and Z. Xie, “Seml: A Semantic LSTM Model for Software Defect Prediction,” *IEEE Access*, vol. 7, pp. 83812–83824, 2019, doi: 10.1109/ACCESS.2019.2925313.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

[5] D. Chen, X. Chen, H. Li, J. Xie, and Y. Mu, "DeepCPDP: Deep Learning Based Cross-Project Defect Prediction," *IEEE Access*, vol. 7, pp. 184832–184848, 2019, doi: 10.1109/ACCESS.2019.2961129.

[6] T. M. Phuong Ha, D. Hung Tran, L. T. My Hanh, and N. Thanh Binh, "Experimental Study on Software Fault Prediction Using Machine Learning Model," in *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, Oct. 2019, pp. 1–5, doi: 10.1109/KSE.2019.8919429.

[7] S. Wang, T. Liu, J. Nam, and L. Tan, "Deep Semantic Feature Learning for Software Defect Prediction," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1267–1293, Dec. 2020, doi: 10.1109/TSE.2018.2877612.

[8] R. Harrison, S. Counsell, and R. Nithi, "An overview of object-oriented design metrics," in *Proceedings Eighth IEEE International Workshop on Software Technology and Engineering Practice incorporating Computer Aided Software Engineering*, Jul. 1997, pp. 230–235, doi: 10.1109/STEP.1997.615494.

[9] M. Wen, R. Wu, and S.-C. Cheung, "How Well Do Change Sequences Predict Defects? Sequence Learning from Software Changes," *IEEE Transactions on Software Engineering*, vol. 46, no. 11, pp. 1155–1175, Nov. 2020, doi: 10.1109/TSE.2018.2876256.

[10] Y. Qiu, Y. Liu, A. Liu, J. Zhu, and J. Xu, "Automatic Feature Exploration and an Application in Defect Prediction," *IEEE Access*, vol. 7, pp. 112097–112112, 2019, doi: 10.1109/ACCESS.2019.2934530.

[11] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol. 58, pp. 388–402, Feb. 2015, doi: 10.1016/j.infsof.2014.07.005.

[12] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, vol. 385, pp. 100–110, Apr. 2020, doi: 10.1016/j.neucom.2019.11.067.

[13] C. Jin and S.-W. Jin, "Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization," *Applied Soft Computing*, vol. 35, pp. 717–725, Oct. 2015, doi: 10.1016/j.asoc.2015.07.006.

[14] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," *Information and Software Technology*, vol. 62, pp. 67–77, Jun. 2015, doi: 10.1016/j.infsof.2015.01.014.

[15] T. Shailesh, A. Nayak, and D. Prasad, "Performance Prediction of Configurable softwares using Machine learning approach," in *2018 4th International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Mangalore, India, Sep. 2018, pp. 7–10, doi: 10.1109/iCATccT44854.2018.9001957.

[16] Y. Tamura, M. Matsumoto, and S. Yamada, "Software Reliability Model Selection Based on Deep Learning," in *2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)*, May 2016, pp. 1–5, doi: 10.1109/ICIMSA.2016.7504034.

[17] R. Malhotra, L. Bahl, S. Sehgal, and P. Priya, "Empirical comparison of machine learning algorithms for bug prediction in open source software," in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, Mar. 2017, pp. 40–45, doi: 10.1109/ICBDACI.2017.8070806.

[18] H. D. Tran, L. T. M. Hanh, and N. T. Binh, "Combining feature selection, feature learning and ensemble learning for software fault prediction," in *2019 11th International Conference*

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 14615 - 14635
Received 25 April 2021; Accepted 08 May 2021.

on Knowledge and Systems Engineering (KSE), Oct. 2019, pp. 1–8, doi: 10.1109/KSE.2019.8919292.

[19] A. Souri, A. S. Mohammed, M. Yousif Potrus, M. H. Malik, F. Safara, and M. Hosseinzadeh, "Formal Verification of a Hybrid Machine Learning-Based Fault Prediction Model in Internet of Things Applications," *IEEE Access*, vol. 8, pp. 23863–23874, 2020, doi: 10.1109/ACCESS.2020.2967629.

[20] E. Elahi, S. Kanwal, and A. N. Asif, "A new Ensemble approach for Software Fault Prediction," in *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, Jan. 2020, pp. 407–412, doi: 10.1109/IBCAST47879.2020.9044596.

[21] J. Ge, J. Liu, and W. Liu, "Comparative Study on Defect Prediction Algorithms of Supervised Learning Software Based on Imbalanced Classification Data Sets," in *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Jun. 2018, pp. 399–406, doi: 10.1109/SNPD.2018.8441143.