FRAUD DETECTION ON BANK PAYMENTS USING MACHINE LEARNING

Kammari Santhosh
Atal Bihari Vajpayee-Indian Institute of Information Technology and Management
Gwalior,474015, India
imt_2018043@iiitm.ac.in

Pinku Ranjan
Atal Bihari Vajpayee-Indian Institute of Information Technology and Management
Gwalior,474015, India
pinkuranjan@iiitm.ac.in

Arun Kumar
Atal Bihari Vajpayee-Indian Institute of Information Technology and Management
Gwalior,474015, India
arunkumar@iiitm.ac.in

Somesh Kumar
Atal Bihari Vajpayee-Indian Institute of Information Technology and Management
Gwalior,474015, India
somesh@iiitm.ac.in

***Abstract*—The importance of software systems and their impact on all sectors of society is undeniable. Furthermore, it is increasing every day as more services get digitized. This necessitates the need for evolution of development and quality processes to deliver reliable software. For reliable software, one of the important criteria is that it should be fault-free. Reliability models are designed to evaluate software reliability and predict faults. Software reliability is always an area of interest in the field of software engineering. Prediction of software reliability can be done using numerous available models but with the inception of computational intelligence techniques, researchers are exploring new techniques such as machine learning, deep learning, etc. to develop better prediction models. In the current study, a software reliability bug classification model is developed using a deep learning technique . It was proven that the cost of fixing errors escalates as a project moves through its life cycle in an exponential fashion. Identifying buggy classes, as soon as they are committed to the Version Control System, would have a significant impact on reducing such cost. Mining in software repositories is a growing research area, where innovative techniques and models are designed to analyze software repositories data and uncover useful information that can help in identification of software bugs. Previous studies showed that Deep Learning has achieved remarkable results in many fields and it keeps evolving. In this paper, experiments are carried out to study the effect of feature selection on the performance of bug prediction models . The primary contribution of this paper is to extend the work in by considering larger datasets, feature tuning by applying feature selection techniques and considering the use of Deep Learning for the classification**

***Index Terms*—Enterprise Modeling, Digital Transformation, Instant Payment., Fallacious Transaction, Online Shopping**

I. INTRODUCTION

The majority of software organizations are small to medium enterprises that experience limited resources and strict deadlines. Under these limitations producing high quality products with the lowest cost becomes very challenging . Testing and bug fixing phases are used to maintain the product quality and it was shown that the escalation of the cost to fix errors as a project moves through its life cycle increases in an exponential fashion.There is no doubt that identify buggy classes as soon as they are committed to the Version Control System (VCS) would have a significant impact on reducing the fixing cost, and hence the product cost.

Reliability is an essential and one of the most critical aspects of a software product and it is also one of the major attributes to determine software quality. Software reliability can be described as its ability to perform its intended functions accurately and successfully. Regular checks during software development ensure the prevention of faults which can further lead to failure and might incur huge efforts to correct or recover if detected later. Therefore, reliability prediction is an important aspect of any software development approach. For reliable software, it is important that it should be fault-free. The current study uses a deep learning-based technique for software reliability prediction due to its potential to predict high accuracy on the huge amount of unstructured or unlabeled data.early fault prediction using deep learning models helps to improve the reliability of the software.

Experiences are built up over time with the development of the human brain through many stages. A developing neuron is as similar as a plastic brain. To adapt with the surrounding environment the developing nervous system has the property of plasticity. Plasticity appears to be essential to the functioning of neurons as information processing units in the human brain. Similarly this same thing happens with neural networks made up of artificial neurons. A neural network is a machine that is designed to model the way in which the brain performs a particular task. To achieve good performance, neural networks should have a massive interconnection of simple computing cells referred to as “neurons” or “processing units”. Neural networks perform essential computations through a process of learning.

Deep learning is a subset of machine learning algorithms that are built on Artificial Neural Network (ANN). Neural networks are computational systems that respond to external inputs with dynamic state changes and try to determine under-

lying relationships within a dataset. ANN with two or three layers is a basic neural network and the neural network with more than three layers is considered as a deep learning concept . The label deep was inspired y the number of processing layers that data must pass through. Deep learning advances have resulted in the development of neural networks with more complexity to generate more powerful learning abilities. The deep learning model takes an input and performs a step-by-step nonlinear transformation and then uses the learnings to generate a statistical model as output. The model continues these iterations until the output is accurate enough. Due to the data-hungry nature of deep learning algorithms and increased dataset size, complex problems can be easily solved more accurately and efficiently.

Deep learning integration into Software Engineering tasks has become increasingly popular among software developers and researchers these days. Deep learning assists SE experts in extracting requirements from natural language text, generating source code, and predicting software faults for typical SE tasks. Deep learning in SE has increased the interest of both the SE and AI experts. The choice of the deep learning model has been determined because of its ability to automatically capture and learn the discriminative features from data, which results in an improved reliability prediction model. This research will open the road for other deep learning approaches to be used in fault prediction. So, that software engineers will be able to better predict the likelihood of faults which results in greater resource use, risk management and better quality control.

II. LITERATURE REVIEW

The use of Computational Intelligence (CI) in the field of software engineering is expanding nowadays. It can be witnessed by the huge research work undergoing and still being carried out by various researchers. Some important research work related to software reliability prediction is filtered and studied to conduct the current work

Costa et al. [**?**] proposed a hybrid approach which used both genetic algorithms and evolutionary neural networks for improving the reliability prediction. In this approach, a genetic algorithm is used to analyze the number of neurons in each layer of ANN.

The use of hybridization became prominent in the field of predicting software reliability. Another study by Pai and Hong [**?**] experimentedfor predicting software reliability.However, the authors suggest exploring other searching techniques for improving the results. Li et al. [**?**] used the Adaboost technique based on machine learning which combines weak predictors into a single predictor to improve prediction accuracy and the results are verified using two case studies.

Similarly, Roy et al. [**?**]a proposed neuro-genetic algorithm in which ANN is trained using backpropagation and further the weights of the network are optimized . Further the results are compared with traditional methods and good results are obtained by the model.

Then, researchers focused more on machine learning and deep learning methods. Malhotra [**?**] reviewed various machine learning techniques for software fault prediction, performance is assessed and compared with statistical techniques. The study proved that machine learning technique models predict software fault better than traditional models, but these techniques are still limited.

Wahono [**?**] proposed three influential frameworks i.e., Lessmann et al., Menzies et al., and Song et al. by combining Machine Learning (ML) classifiers for predicting software defects and improving the accuracy but these frameworks are not able to handle noisy data.

Clemente et al. [**?**] developed a predictive model using a deep learning technique that predicts security bugs with more accuracy (73.50%) as compared to machine learning techniques. Moser et al. [**?**] made a comprehensive analysis of the efficiency of change metrics and static code attributes for defect prediction. Graves et al. [**?**]The change history contains more useful information than could be obtained from the product (size and structure) metrics

L. Madeyski and M. Jureczko, [**?**]The number of distinct commits(NDC) and number of modified lines(NML) metrics were valuable with regard to bugs. NDC metric regards the number of different developers who have changed a given class. When several people are involved in the process of development, it is always probable that they will misunderstand one another's intentions and overwrite the modifications made by their colleagues, which may, in turn, result in defects. metrics significantly improved the efficiency of prediction and, therefore, are worth taking into account while building the defect prediction models.

Otoom, A. F. et.al [**?**] said that the primary goal, according to the researchers, is to create an intelligent classifier adept of predicting the severity. Train a number of discriminative models based on this data that will be used for automated bug report labeling and severity prediction. When it comes to automatic marking, the precision is about 91 percent.

Ni, Z. et.al [**?**]aimed at spontaneously classifying defect into their root cause categories by using the associated connection among defect fixes and defect causes. To begin, the code-related bug classification criteria in terms of the bug's cause is defined. The results of the experiment show that there is an empirical connection between the bug fix and the cause of historical bugs, and that the prophecy is right.

Prabha, C. L., et.al [**?**] exposed that while contributing to industrial results the observable outcomes are provided by software defect prediction . The developers will identifying bugs from development faults predicting defective code areas. Using the parameters precision, accuracy etc., a research analysis is conducted and results are compared.

K. Han, [**?**]a novel unsupervised feature selection method which could jointly learn a self-representation autoencoder model and the importance weights of each feature. The autoencoder nonlinearly represent each feature using all the features with different weights. By minimizing the reconstruction error and the group sparsity regularization simultaneously'

we obtain a subset of observed features which can preserve intrinsic information of the original data. Experimental results on several benchmark datasets validate the superiority of our methods over other unsupervised feature selection methods.

Analyzing the bug fix corresponding to software bugs is an important means for developers to find the cause of bugs. At the same time, the source code of historical fixed bugs contains a wealth of knowledge about bugs. Zehn Ni [**?**]proposed to extract and represent the knowledge in the source code with a manual labeling for this dataset using the modification categories. Using the as input of different classification models, we can automatically classify the bugs into bug cause categories. The experimental results show that there does exist a relation between the bug fix and the cause of bugs, and the representation model can improve the ac- curacy of bug classification.

III. SYSTEM ARCHITECTURE

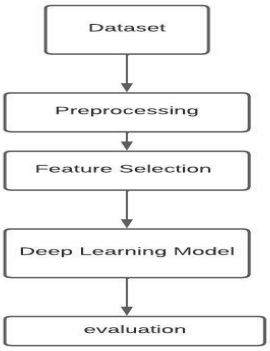

Fig. 1. System Architecture

The system architecture includes dataset preprocessing ,After preprocessing feature selection is to be done.The output of feature selection is given as input to the Deep learning model and a comparative study performed. The proposed bug prediction model is based on three main steps: Attributes(Metrics) retrieval, Feature selection and Bug severity classification model as depicted in Fig.3.1

IV. MODELING

A. Deep learning Model

Deep learning algorithms has shown extraordinary results in various fields in last few years. The most significant distinction between deep learning and regular machine learning is how well it performs when data scales up. Deep learning techniques do not perform well when the dataset is tiny. This is due to the fact that deep learning algorithms require a vast quantity of data to properly understand it. I will be using combine models of CNN and BiLSTM for software bug severity classification

B. CNN

Convolutional Neural Networks (CNN/ ConvNet) are a type of neural network utilised extensively in computer vision. It It is made up of number of hidden layers, as the name indicates. Convolutional layers, pooling layers, fully connected layers, and normalising layers are common hidden layers in CNNs. The collection of the Visual Cortex inspired the construction of a Convolutional neural network, which is very similar to the connection network of Neurons in the Human Brain. Single neurons respond to certain functional reactions exclusively in the Receptive Field, a small area of the visual field. A group of similar fields may be stacked on top of each other to cover the entire region.

C. Bidirectional LSTM

BiLSTMs are an update to unidirectional LSTMs which can increase the model performance on time sequence data. BiLSTM is a sequence model which is consist of two LSTMs. One for the input of forward direction and another one for backward direction that is from the past inputs. BiLSTM trains two LSTMs on input. BiLSTMs increase the amount of information to predict the output. It preserves the future information.

D. CNN-BiLSTM

By combining CNN and BiLSTM. CNN uses several building blocks like as convolution layers, pooling layers, and fully connected layers to learn spatial hierarchies of information automatically and adaptively through backpropagation. And inputs will be run in two directions, one from the past to the future and the other from the future to the past, when you use bidirectional LSTM. This strategy differs from unidirectional in that information from the future is stored in the backward LSTM, while employing the two hidden states together, information from the past and future may be retained at any point in time this will help in learning long range temporal features from the dataset.

E. Model Architecture

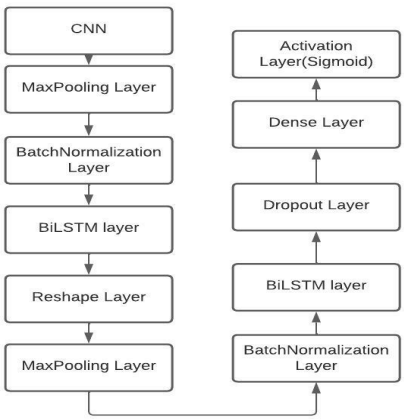

Fig. 2. Model Architecture

An ensemble deep learning model which includes CNN and BiLSTM with other layers like Normalization layer and Reshape layer.

- The reason for using CNN layer and Max pooling Layer is that we know that CNN is best for spatial features and parameter sharing. Parameter sharing helps in reducing the features. Max pooling layer used to know the relative features. Which helps in reducing training time by great factor and decrease over fitting.
- After that there is a Batch normalization layer which does the normalization between the layers to increase the training speed. Next to Batch normalization layer there is Reshape layer which will change the output of the previous layer for the next layers that is BiLSTM layers.
- BiLSTM trains two LSTMs on input it done in order to learn the forward and backward data . I have used pair of BiLSTM layer , between these two layers there is a Max pooling layer remove the features with less relevance and a Batch Normalization layer for normalization of output from the previous layer to increase the performance and training speed.
- After these BiLSTM layers a fully connected dense layer is dense layer is applied as an output layer with the dropout layer next to it. The reason of using dropout layer is to reduce over fitting model have also used Max pooling layer for over fitting reason for this is that generally when CNN and LSTMs are used in combination there are very high chances of over fitting

F. RESULTS

With the use of all four algorithms ,Comparing accuracies of the four Models(from Table 1) to get the better understanding of which algorithm works better for a given dataset.

TABLE I
ACCURACY MATRIX

MODEL	ACCURACY
Logistic Regression	91.38
K-Nearest Neighbor	92.66
Decision Tree Classification	96.11
Random Forest Algorithm	96.64

G. CONCLUSION

We can conclude that the amount of time required for the result and the cost for manual testing and the limited availability of test kits. So we need a technique that provides the work faster and reduces the cost to check. Still, this model can return good accuracy and can be further improved.

H. ADVANTAGES

- To find the deep and implicit similarities in data set
- All possible fraud scenarios are automatically detected
- decreased the total number of verification steps and measures
- real-time processing

[1], [2], [3], [4], [5], [6]

REFERENCES

[1] S. Delecourt and L. Guo, "Building a robust mobile payment fraud detection system with adversarial examples," in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 103–106, IEEE, 2019.

[2] T. Alquthami, A. M. Alsubaie, and M. Anwer, "Importance of smart meters data processing – case of saudi arabia," in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–5, IEEE, 2019.

[3] V. Jain, M. Agrawal, and A. Kumar, "Performance analysis of machine learning algorithms in credit cards fraud detection," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 86–88, IEEE, 2020.

[4] S. Khatri, A. Arora, and A. P. Agrawal, "Supervised machine learning algorithms for credit card fraud detection: A comparison," in *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pp. 680–683, IEEE, 2020.

[5] O. Adepoju, J. Wosowei, S. lawte, and H. Jaiman, "Comparative evaluation of credit card fraud detection using machine learning techniques," in *2019 Global Conference for Advancement in Technology (GCAT)*, pp. 1–6, IEEE, 2019.

[6] A. Thennakoon, C. Bhagyani, S. Premadasa, S. Mihiranga, and N. Kuruwitaarachchi, "Real-time credit card fraud detection using machine learning," in *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pp. 488–493, IEEE, 2019.