# AI-DRIVEN FRAUD DETECTION: SECURING BANKING TRANSACTIONS

Mr.S Sakthi
Asst .Professor
*School of CSE and IS (of Affliation)*
*Presidency University (of Affliation)*
Bengaluru,560064,India
sakthivijayan80@gmail.com

Vijita Narayan Nayak
Master of Computer Applications
*School of Information Science (of Affliation)*
*Presidency University(of Affliation)*
Bemgaluru,560064,India
vijitanayak14@gmail.com

***Abstract*—The importance of software systems and their impact on all aspects of society cannot be denied. Moreover, as increasingly more services are being digitalised, so it is increasing day by day. To ensure reliable software, this necessitates the transformation of development and quality processes. One of the most significant requirements of reliable software is that it should be error-free. Software reliability is evaluated and defects are anticipated based on reliability models. Software reliability is of interest to the field of software engineering at all times. While numerous models are present for predicting software reliability, researchers have started exploring new ways such as machine learning, deep learning, etc. because computational intelligence methods were developed to design better prediction models. A software reliability bug model has been designed using a deep learning method in the current study. It has been shown that the price of error correction rises exponentially with the progression of a project through the life cycle. Would often be the case that establishing the bug classes are the quickest right after the version control system has an issue. Data mining software repositories is an emerging domain of study that is concentrating on generating new methods and models to discover and extract useful information from the repositories to help with the discovery of bugs. Deep learning has some remarkable results in several areas according to previous studies. Specifically, the efficacy of the bug prediction model is evaluated in this work by varying the number of features. This paper is mainly about the extension of the initial work, with more extensive datasets, a better search for the features, and the use of Deep Learning for classification.**

***Index Terms*—Software systems, Malicious Transaction Software Defects, Prediction .**

## I. INTRODUCTION

A large part of software companies are not huge conglomerates but mostly small to medium-sized enterprises with tight schedules and minimal resources. It distresses greatly to produce the best products using least money under such limits. The testing and bug fixing stages are there to keep up the product quality and it was argued that the cost of error fixing increases expotentially as a project goes through different stages of its life cycle.

Without any doubt, being able to identify defective classes once they get entered into a Version Control System (VCS) would surely lead to a decreasing fix cost, consequently, a very important reduction in the overall price of the product..

Speaking of software quality, reliability complex and vital parts. It is the top characteristic of a software that determines its quality. Software reliability is the ability of the system to perform the necessary operations accurately and productively. Regular checks during software development are helpful to avoid the appearance of faults, so that failures are less likely to occur, and the necessity of correction or recovery of faults after a long period of time is partly eliminated. Consequently, the prediction of reliability is the main driver of the development of software .By eliminating any errors, the software reliability will be ensured without doubt .This work stands out for its focus on the use of a deep learning approach in software reliability prediction, as deep learning can bring about high accuracy even in the case of unstructured and unlabeled data. The software's reliability is improved when deep learning models can detect failures at an early stage.

Experiences are generated by the human brain through various stages of its development. A new neuron in the brain has as much flexibility as a piece of plastic. So as to be able to respond to the varying environment, the growing nervous system has the property of being plastic. It is said that the plasticity is necessary for the functioning of neurons as the human brain's information processing units. Analogously, the same thing is true of artificial neurons that make up neural networks. A neural network is a computer designed to mimic the way the brain accomplishes tasks. To perform the task effectively, neural networks should be made up of numerous simple computing cells, known as “neurons” or “processing units.” Neural networks carry out the major work of cognition through the learning process.

Deep learning is a collection of machine learning methods that is based on the subset of these algorithms called as Artificial Neural Network (ANN).

ANN that have only two or three layers are just simple neural networks while networks with more layers than three are at the core of deep learning.

The term deep was decided because of the many stages of processing that the data has to be transmitted through. With the progress of deep learning, we now have neural networks of higher complexity to support more robust learning capabilities. The deep learning model receives an input and then does a step-by-step non-linear transformation to produce the learnings and then use these learnings to create a statistical model as an output. The model makes these iterations go on until it has gained the most accurate outcome. As a result of the 'data- guzzling' factor of the deep learning algorithms and the increasing size of the data set, difficult problems can now be solved very accurately and very efficiently.

The increasing number of AI experts using deep learning to solve Software Engineering tasks is an illustration of successful integration. There are a lot of examples of integrations of deep learning with software engineering, resulting from the effort of a horde of software developers, and researchers. Deep learning can be utilized to perform such tasks as extracting requirements from natural language, generating source code, predicting software faults, and doing this effectively in SE. Deep learning in SE is advantageous to both the AI and the SE communities. The choice of the deep learning type is undertaken grounded upon the direct capacity of the chosen deep model to take in the most distinguishing features from the data and then build a more reliable prediction model. As a kind of a preliminary work of utilizing deep learning in the task of fault prediction, this inquiry leads to the implementation of other deep learning methods in that direction. By then, the software engineers shall have the capability of predicting the fault probability they now have the ability to use their resources more efficiently, manage the risks, and improve the quality of the product.

## II. **LITERATURE REVIEW**

It is observed during the present times that there has been a rise in the use of Computational Intelligence (CI) in software engineering. We can see that the existence of a large number of researchers and scientists who are involved in doing research through writing papers is an indication of this trend. Our reports that focused on software reliability prediction discuss many researchers' studies and are a necessary part of our analysis. The authors proposed a method that uses a combination of the genetic algorithm and the network via a hybrid approach that improved the reliability prediction. This work used also genetic algorithms in the study of the examination of the number of neurons in each layer of ANN.

A different research job on software reliability prediction is done by Pai and Hong.They analyzed Adaboost technique which is a very good machine learning form. It takes a few bad predictors and then combines them to form a single strong predictor which will give the correct accuracy.

The authors also carried out checks on the obtained results with two case studies. They were using the proposed ANN model and the adaptation of the network weights when necessary for the monitoring of the ANN (some phrases were not clear). In addition, they compare the algorithmic solution with the traditional methods where the new algorithm shows maximum performance.

Furthermore, the swarm of researchers was attracted by deep learning and machine learning who discovered the methods later. In a reviewed paper, Malhotra [?] The papers gathered and documented the data, showed several machine learning methods that could be used for software fault prediction, and compared the performance of various methods using statistical tests. While these techniques are still in development, the research at hand confirmed that machine learning models were superior to traditional models for the purpose of software fault assessment.

To detect software bugs and enhance precision in the results, Wahono's research [?] came up with three strong frameworks—Lessmann et al., Menzies et al., and Song et al.—which consisted of Machine Learning (ML) classifiers. The problem with these suggested frameworks was that they could not work with noisy data. They have also introduced a deep learning model that can better predict security system flaws with a higher accuracy rate (73.50%) than any of the other machine learning methods in the report back (Clemente et al.(13). Moser et al.'s research [?]proposed an inquiry about how static code metrics and change metrics influenced defect prediction.
M. Jureczko and L. Madeyski, [?]The metrics that were most helpful in terms of bugs were unquestionably the distinct commits (NDC) and modified lines (NML). The number of distinct developers who have modified the specified class is the subject of the NDC measurement. When multiple people participate in the development process, there is a significant risk that they will misunderstand and overwrite the changes made by their peers, which could lead to defects. These metrics are crucial for software failure models because they predict prediction efficiency using NDC and NML.

According to Otoom, A. F. et al. [?], the authors' primary goal was to develop a strong classifier that could forecast how serious the bugs would be. Several discriminative models that can be used to predict the severity of the bug report will be trained using the data as a basis. The accuracy during automatic marking is approximately 91%.

By using the corresponding relationship between defect fixes and defect causes in an organic way, Ni, Z. et al. sought to classify defects into their root cause categories. All that was the need for a code-related bug classification standard that can tell us the reason for the bug. The study provides the evidence that the idea is correct and there is a definite relationship between the bug repair and the original cause of the faults.

Software defect prediction, according to Prabha, C. L., et al., is carried out to obtain the observed results contributing to the industrial results, while developers will identify bugs from development faults to predict the defective code areas. The research analysis is carried out and the results are matched by using the parameters precision, accuracy, etc.

K.Han is a novel unsupervised feature selection technique that uses the weight of each feature that can be updated separately to create a self-executable representation of each unsupervised characteristic. A fictitious example Every feature is represented nonlinearly by the autoencoder using varying weights for each feature.

Our methods proved superior to other unsupervised feature selection methods in a series of experiments on benchmark the datasets. The results are based on our experiments.

Identifying and solving a bug by understanding the relevant software bugs is the most effective method for developers to diagnose the trouble of those bugs. Moreover, the historical bug fixes' source code carries a lot of information on the bugs. Zehn Ni careered in his idea that the information in the source code could be collected and visualized through manual labeling for this dataset using the modification categories. The cause of the bug can be ascertained by performing multiple classifications of the bugs with the help of input to different classification models. The experiment results not only confirm that the bug fix is indeed the cause of the bug, but also reveal that through the use of the representation model, the accuracy of bug classification can be enhanced.

## III. SYSTEM ARCHITECTURE

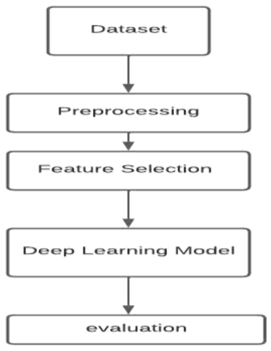

Fig. 1. System Architecture

Dataset preprocessing is part of the system architecture; feature selection follows. A comparative analysis is conducted after the feature selection output is fed into the deep learning model. The three primary steps of the suggested bug prediction model are feature selection, bug severity classification model, and attributes (metrics) retrieval, as shown in Fig.1.

## IV. MODELING

### A. *Deep learning Model*

Deep learning models have shown great success in recent years in resolving issues across various fields. The percentage of data that deep learning can handle effectively sets it apart from other machine learning techniques. Deep learning methods are not efficient in dealing with small datasets. This is due to the fact that deep learning algorithms require extensive data in order to understand it correctly. I have decided to employ CNN and BiLSTM models for software bugs severity classification.

### B. *CNN*

One common type of neural network in the field of computer vision is the Convolutional Neural Network (CNN/ConvNet). Just like its name suggests, it is made up of a number of hidden layers. The decomposition of CNN hidden layers is usually carried out by normalizing, fully connected, and convolutional layers.The authors developed the CNN (Convolution Neural Network) with one hand using the functioning of the visual cortex and with the other hand by the ability of CNNs to be the most important similarity between the human brain and CNNs. Only in the Receptive Field, which makes up a tiny percentage of the visible field, are individual neurones in charge of particular functional reactions. A group of these fields may overlap, thus encompassing the entire region.

### C. *Bidirectional LSTM*

An improvement on unidirectional LSTMs, bidirectional LSTMs can improve model performance on time sequence data. A sequence model called BiLSTM is made up of two LSTMs. One is for the forward direction input, and the other is for the backward direction, which is based on previous inputs. BiLSTM uses input to train two LSTMs. BiLSTMs use more data to forecast the result. It keeps the information for later.

### D. *CNN-BiLSTM*

CNN and BiLSTM came together. CNN uses backpropagation to automatically and adaptively learn the spatial hierarchies of the data through each of its convolutional, pooling, and fully connected layers. Therefore, the inputs will move in two directions when using a bidirectional LSTM. This method may encourage the two hidden states to carry information from the past and future at any time, allowing them to do so throughout the entire sentence. When you use bidirectional LSTM, a portion of the information will be passed from the later stages of the network to the early stages through the backward LSTM. Learning long-range temporal features from the dataset will benefit from this.

### E. *Model Architecture*

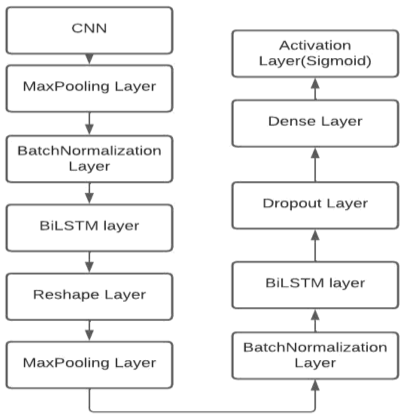

Fig. 2. Model Architecture

CNN, BiLSTM, and additional layers like the Normalisation and Reshape layers make up this ensemble deep learning model.

- The reason for using CNN because CNN is the best at sharing parameters and spatial features, we have included CNN and Max pooling layers here. To cut down on features, however, parameter sharing is used. By using max pooling to extract the key features from the originality, training time can be drastically reduced, which will reduce overfitting.
- The batch normalization layer follows, taking care of layer-to-layer normalization in a quicker way for training. The Reshape layer, which is located after the Batch normalization layer, will transform the output from the previous layer into the BiLSTM layers.
- The BiLSTM layers are combined with the fully connected dense layer, which serves as the output layer, and then the dropout layer that follows it, makes the architecture for the models, employed in image understanding, become more emphasized. The dropout layer is there to protect the model from overfitting. To tackle the problem of model ovefitting and the matters of CNN and LSTMs, they were generally combined and these two layers are all of which they have also included a Max pooling layer in order to prevent this.
- BiLSTM is a model that machine learning trains to learn the input and the feedback data source is two sets. In my experiment, I have used two BiLSTM, a Max pooling layer, a Batch Normalization layer, and other techniques to reduce the dimension of the data set and speed up training process to perform better. In my experiment, I used two BiLSTM with a Max pooling layer allowing me to reduce the number of dimensions of the feature space to avoid the effect of irrelevant features and a Batch Normalization layer for the normalization process of the output data set.
- As a result, the CNN layers perform preprocessing at the model's initialization point and are crucial components that should be used to locate objects in an image. These layers use filters to examine the input and identify important elements, such as textures or edges. Max pooling is a technique used after CNN layers to reduce the initial dimensions of the data. It also speeds up the model and prevents overfitting by removing features that aren't relevant.
- Following data reshaping, BiLSTM layers are added and used to capture both preceding and following context in a sequence, taking into account the features' sequence order in a useful manner. The model can understand longer-term patterns thanks to the use of two BiLSTM layers, and the Max pooling between them helps choose pertinent information from a large amount of data. The most accurate and effective results are obtained by combining a CNN with a BiLSTM to create a model with both spatial and temporal properties.

## F. RESULTS

By using all four algorithms, it will be easier to compare the accuracy of the four models (from Table 1) to get the better understanding of which algorithm works better for a given dataset.

TABLE I

| MODEL | ACCURACY |
|---|---|
| K-Nearest Neighbor | 94.42 |
| Logistic Regression | 91.94 |
| Random Forest Algorithm | 93.33 |
| Decision Tree Classification | 92.93 |

## G. CONCLUSION

As a result, there are three things that were of utmost importance to be identified: the time that it took in order to get the results, the manual testing expenses, and the deficit of available kits. What was necessary in this case would be to take up an alternative that not only is able to perform the same task quickly but also is not expensive. If the above model can ensure the high quality too, it will be extremely advantageous.

## H. ADVANTAGES

The data set's deep and implicit similarities can be found all potential fraud scenarios are automatically identified, the overall number of verification steps , measures is reduced and processing is done in real time.

## REFERENCES

[1] Y. Zhu, C. Wang, and M. Sun, “Neural-SMOTE: A hybrid deep learning approach to imbalanced credit card fraud detection,” arXiv preprint arXiv:2405.00026, 2024.

[2] L. Xiang, J. Li, and H. Zhao, “Graph neural fraud detection with temporal gated attention,” arXiv preprint arXiv:2412.18287, 2024.

[3] Q. Zheng, T. Huang, and Y. Liu, “Boosted ensemble model for real-time financial fraud detection,” arXiv preprint arXiv:2406.04658, 2024.

[4] J. Yu, F. Zhang, and R. He, “Credit card fraud detection using transformer-based deep learning models,” arXiv preprint arXiv:2406.03733, 2024.

[5] A. Ali, R. Muneeb, and T. Khan, “Enhancing fraud detection with GAN-based synthetic data augmentation,” in Proc. Int. Sci. Conf. Knowledge & Utilization (ISCKU), vol. 18, pp. 76–81, 2024.

[6] K. Praveen, S. Natarajan, and D. Kumar, “Machine learning for fraud detection and financial risk assessment in banking,” J. Innov. Eng. Res., vol. 6, no. 3, pp. 193–200, 2024.