Journal of Computational Science and Intelligent Technologies
eISSN: 2582-9041
DOI: 10.53409/MNAA/JCSIT/e202102030108
Type: Research Article

Classification of Bugs using Machine Learning Algorithms

Aishwarya Jayagopal[1], Kaushik R[2], Arun Krishnan[3], Ramesh Nalla[4], and Suresh Ruttala[4]

[1]Independent Researcher, Palakkad, Kerala, India
[2]Independent Researcher, Bengaluru, Karnataka, India
[3]Independent Researcher, Kozhikode, Kerala, India
[4]Independent Researcher, Hyderabad, Telangana, India
[*]Corresponding Author: Aishwarya Jayagopal. Email: jayagopalaishwarya@gmail.com
Received: 21 March 2020; Accepted: 10 July 2021

Abstract: DevOps is a method used to automate the process between the development team and the IT team through which they can develop, test, and release their software. Bugs during this stage slow the entire release cycle. To overcome this, Machine Learning and Deep Learning Algorithms are used to analyze and arrive at the possible cause of the bug. This reduces the dependency on the developers and, in turn, speeds up the release cycle. The bug dataset is fed to various classification algorithms like CNN, Random Forest, Decision Tree, SVM, and Naïve Bayes for bug classification. Based on the experimental results, it can be observed that Convolutional Neural Networks, a deep learning technique, outperformed all the other approaches used. Furthermore, it was observed that Naïve Bayes, a probabilistic classifier generally preferred for text classification, performed poorly with the bug dataset used in this paper. Ensemble methods like a Decision tree and Random Forest performed better on this dataset.

Keywords: DevOps; Machine Learning; Deep Learning; CNN; bugs

1 Introduction

The DevOps paradigm has gained much popularity in the Information Technology world today since it enables teams to quickly deliver stable, reliable products. It ensures swift release cycles, thereby improving the maintenance and upgrade timelines. DevOps combines agile with automation and continuous delivery, promoting efficiency between development and operations teams and providing faster innovations and superior deliverables to customers and businesses. Bug Tracking Systems (BTS) play an important role here as it emphasizes better quality faster, thus rendering better service and customer satisfaction. The BTS systems allow for efficient communication and collaboration between the reporters of the bugs and the respective feature developers by posting comments and attachments, giving more information on the errors, and promoting speedy resolution. In addition, these systems have features that provide detailed information about team members' efforts to analyze and solve a bug reported, statistics on the number of bugs reported, and many more items to debug.

However, even with the combination of the above two aspects, that of DevOps and BTS, there are still gaps in smooth product delivery. The main cause is that the bug counts for large enterprise projects are generally high, and sorting and contacting the respective developers for a fix is still manual. This increases the resolution time for each bug raised. Since there is human intervention in the bug assignment and bug

fixes process, there are high chances of errors, which further delay the bug resolution, affecting the entire release cycle. This is an area wherein Machine Learning, and Deep Learning algorithms can be used to classify the bugs reported and suggest a fix to the tester, thus removing the manual overhead. This paper has used a system developed for bug classification with Atlassian JIRA [1] as the BTS and the bug data generated from the DevOps tool named Chef [2]. This paper suggests a two-layered approach, wherein the algorithms first detect the nature of the bug in the first layer and then, based on its nature, identify the plausible cause in the second layer. Various supervised learning algorithms were experimented with for classifying data, and their performance was observed and analyzed. It was observed that the Convolutional Neural Networks (CNN) outperformed others. This paper is segmented into five parts. Introduction falls under Section 1, followed by Related Work in Section 2. This section focuses on the study done by others in this area. The working of the proposed system is elaborated in Section 3. Section 4 describes the results obtained. The paper concludes with a possible future direction this research can take, as highlighted in Section 5.

2 Related Work

Natural Language Processing, which encompasses text classification, finds extensive application in domains like spam detection, sentiment analysis, etc. Textual data has an immense level of information within it, but extracting relevant data from it is tedious and time-consuming owing to its unstructured nature. CNN focuses on obtaining patterns from the input textual data, passing them through various filters, to extract feature maps. These are then passed through multiple hidden layers for further feature extraction, and the final output is obtained from the output layer of the neural network. Research has been done using the reports generated from bug-tracking systems to classify the bugs into various categories. The research undertaken by Diksha Behl, Sahil Handa, and Anuja Arora proposes using TF-IDF along with the Naïve Bayes approach [3] to classify bugs into security and non-security categories. A vocabulary of bugs was created from all the bug reports, and each report was converted into a vector, indicating the occurrence of certain words in the report. Based on the weighted values of each complete word vector, the report was classified as a security or non-security bug.

The researchers of [4] have worked on a bug reports classification system. This system uses N-gram Inverse Document Frequency to derive phrases from the bug report to categorize the bug report as a valid bug. These features were inputted into Logistic regression and Random Forest classification models. This approach was compared with a topic modeling approach used for similar bug classification. The N-gram IDF approach was found to be more accurate in both models. Tao Zhang and Byungjeong Lee, in their paper [5], have described a method to detect identical bug reports based on bugs reported on Bugzilla. They made use of bug rules and text-based similarities to identify duplicates. Taxonomy classification and hierarchical clustering algorithms were used to classify the bug into the appropriate category. The paper by Neelofar *et al.* [6] provides a method to classify bugs and compares feature extraction methods of TF-IDF and Chi-Square algorithms. This research concluded that the Chi-square algorithm performed better than TF-IDF in prediction accuracy. The idea behind the classification was that correct classification of bugs can help the bug triage team to assign the reported bug to the right developer. The approach proposed by Lin Tan and co-authors in their research paper [7] seeks to analyze the bug reported and create a patch based on its root cause. R2Fix classifies the bug, obtains the parameters used in the source code based on the report from the classifier, and uses these patterns to generate a patch. This paper seeks to improve the bug classification aspect of the previous studies and aims to classify the bugs to arrive at the root cause of the issue. This is achieved through two layers of classification. The subcategory thus arrived at makes it easier for the developer to provide a solution.

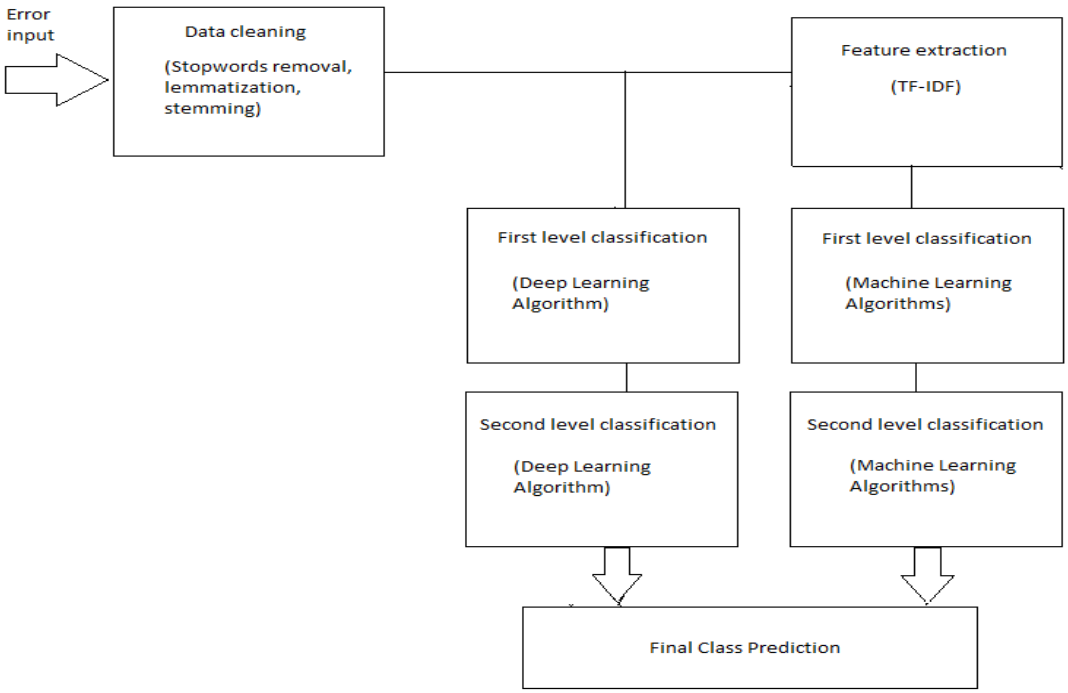

Figure 1: Architectural diagram of the proposed system

3 Proposed System

The pictorial representation of the model used in the paper is shown in Fig. 1. Data obtained from Atlassian JIRA, a bug-tracking system, has to undergo pre-processing, after which it will be subjected to various Machine Learning and Deep Learning Algorithms to learn from the existing bugs and classify the new bugs based on its learning. A two-layered classification is done wherein the first layer classification predicts the bug's nature based on this class. A particular sub-class is predicted to give a plausible fix to the tester. The errors collected during the operational time are stored and later used to re-learn the algorithm after the storage reaches a certain threshold.

3.1 Dataset

A Bug tracking system called Atlassian JIRA was used to collect the bug list. The first error level was essentially the output provided by the Chef Orchestration tool. The errors, in CSV format, were used as input to the algorithms. The dataset consisted of 948 samples.

The dataset was categorized into 5 classes, namely:

1. *Appserver Scripting Tool (270 samples)*
   This category deals with app server Scripting Tool related errors.
2. *Middleware (70 samples)*
   This category deals with errors related to Middleware installation, patching, and upgrades.
3. *Machine (315 samples)*
   This category deals with errors arising due to machine configurations, permissions, resource limitation
4. *The compilation (117 samples)*
   This category deals with issues in code quality, like variable declaration issues and syntactical issues.
5. *Application (176 samples)*

This category deals with issues specific to the application deployment and upgrades.

The sub-classification for each of the categories mentioned above is as follows:

- AST
  i) Startup/shutdown – Errors related to service reboot
  ii) Domain – Errors related to the improper domain configuration
  iii) Machine/Properties issues – Errors related to wrong attribute references in property files passed to the AST scripts
  iv) Connection/SSL issue – Errors caused by connection issues to application servers due to improper server configurations
- Middleware
  i) OPatch – Issues obtained during patching of Middleware
  ii) RCU - Issues in creating DB schemas
  iii) Machine - Issues specific to the virtual machine, its configurations, and permissions
  iv) DB issues – Issues related to the database connection, like incorrect credentials, lack of relevant schemas, etc.
- Machine
  i) Missing file – Issues due to lack of required file
  ii) Missing Parent Directories – Issues arising due to the lack of a parent directory to create contents within it
  iii) Rerun/System exit – Issues arising due to manual interrupts
  iv) Permission issue – Issues related to incorrect permissions
  v) Package issue – Issues arising due to unavailable packages in the yum repo
  vi) Connection – Issues due to network and connection
  vii) Timeout - Issues due to lack of resources or command timeout
- Compilation
  i) Missing Declaration – Issues arising because of missing variable declaration
  ii) Missing Attribute – Issues due to lack of variable assignment
  iii) Syntax Errors – Chef-specific syntax errors

Note: Application errors are not sub-classified as they are specific to the applications and not the orchestration process.

Following are a few examples of commonly encountered errors [8] [9]:

1. **Sample error 1:**
   Error Starting server AdminServer: nodemanager.NMException: Exception while starting server 'AdminServer'

2. **Sample error 2:**
   Parent directory /user1/postgresql/9.x/bin does not exist.
   Error executing action create on resource 'template[/user1/postgresql/9.x/bin/postgresql.conf]'
   Chef::Exceptions::EnclosingDirectoryDoesNotExist

3.2 Pre-processing

Text data needs to be pre-processed as it helps in improving accuracy. The textual data was converted to lowercase to ensure case insensitivity. The words in the NLTK stop words corpus (English) were removed along with the other non-alphabetic characters, like numbers and punctuation marks, since they do not contribute to the predictions. Stemming, a process of deriving the word's root form, was also applied.

3.3 Machine Learning Algorithms

3.3.1 Feature Engineering

Feature Engineering is the method of selecting suitable features for the algorithm. This paper has used a method called "Term Frequency-Inverse Document Frequency," known as TF-IDF.

TF-IDF is a numeric measure to judge the significance of a word in the collection. The significance of the word jumps up when the occurrence of a word in the collection offsets the occurrence of the word in the document increases.

$$\text{TF-IDF}(x) = \text{TF}(x) * \text{IDF}(x) \quad (1)$$

Where

TF(x) = (Count of term x present in document) / (Total count of terms in the document).

IDF(x) = $\log_e$ (Total count of documents/Count of documents with term x:- ln(x)).

3.3.2 Classification

This paper compares the performance of a few classification algorithms such as Support Vector Machine, Decision Tree, Naïve Bayes, Random Forest, and Convolutional Neural Networks (CNN). A tree-based model for decision-making is employed in Decision Trees [10]. They yield resource costs, utility, and chance event outcomes as results. The tree is split into edges based on condition/internal node attributes. The last branch, thus obtained when the further split is not possible, is the decision/leaf. The categorized class labels will be the child nodes.

$$\text{Entropy}(M) = -\sum p(M) * \log p(M) \quad (2)$$

$$\text{Gain}(L,M) = \text{Entropy}(L) - \text{Entropy}(L,M) \quad (3)$$

Random Forest [11] uses labeled training data, creates multiple decision trees, and merges them into one, thereby improving prediction accuracy. It is similar to bagging classifiers and decision trees in terms of the hyperparameters used. A subgroup of features determines the node splitting, contributing to the classification. Using separate thresholds for each feature can modify the tree structure and contribute better to the accuracy of the predictions than choosing a common threshold for all features.

Support Vector Machines (SVM)[12] are discriminative classifiers that categorize new samples based on the hyperplane generated during the training phase. In two-dimensional space, the hyperplane is a line dividing the 2D space into two parts corresponding to the classes in the data. A very different perspective to the ensemble model is provided when bagged with other machine learning algorithms.

Another algorithm used for text-based classification is the Naïve Bayes algorithm based on conditional probability [13]. The core part of the algorithm comes from Bayes theorem, which states

$$P(N|O) = (P(O|N) * P(N))/P(O) \quad (4)$$

3.4 Deep Learning Algorithms

The algorithm used under Deep Learning methodology is Convolutional Neural Network, also known as CNN[14]. It is a class of deep, feed-forward artificial neural networks (the connections between the nodes in various layers do not form a cycle). For making minimal pre-processing, it makes use of a variety of multi-layered perceptrons. The animal visual cortex was the source of inspiration for this algorithm. Two important operations in CNN, which can be considered feature extractors, are convolution and pooling. Convolution can be considered as applying a filter over a fixed-size window. Pooling merges the vectors resulting from various convolution windows into a single-dimensional vector. After several max pooling layers and convolutional layers, fully connected layers provide high-level reasoning in the network. Neurons in a fully connected layer are activated via affine transformation, with matrix multiplication and a bias offset.

4 Experimental Results

The approach used in this paper successfully performed a two-stage classification of bugs reported from the execution of Chef cookbooks. Comparing various machine learning techniques and deep learning algorithms found that the deep learning CNN algorithm gives the highest accuracy score. Since the second classification layer depends on the first layer, having a high-accuracy model in the first layer is the most important aspect. Fig. 2 compares various classifiers in the first stage of classification. The graph in Fig. 2 shows that CNN achieves the highest accuracy score of 97.73 percent compared to the other algorithms, whose accuracy score is less than 90 percent for the same data. The naïve Bayes algorithm yielded an accuracy of 85 percent. The decision tree algorithm classified 76 percent accurately compared to the 86 percent of Random Forest. The performance of SVM surpassed that of the other machine learning algorithms with an accuracy of 87 percent.

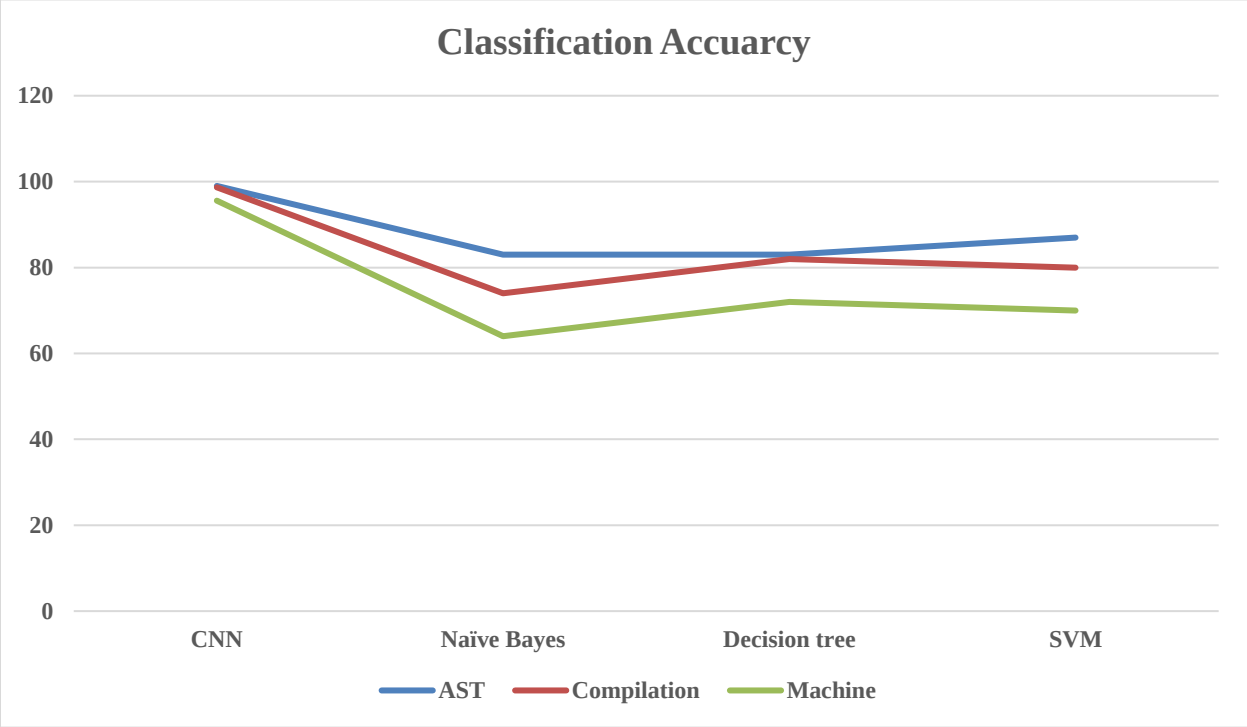

Figure 2: Accuracy, in percentage, for First Level Classification

Even though Naive Bayes is considered a standard approach for text data classification, but it is less accurate when compared with deep learning algorithms. Although convolutional neural networks are traditionally used for image data, it was also found to perform very well on text-based data. From the comparison of accuracy on test data, it was observed that CNNs perform the best on both the first level classification and each sub-classification in the second stage. The results for the same are shown in Tab 1.

Table 1: Accuracy percentages from second level of classification

Algorithm	AST	Compilation	Machine	Middleware
CNN	99	98.67	95.57	98.83
Naïve Bayes	83	74	64	61
Decision tree	83	82	72	70
SVM	87	80	70	75
Random Forest	88	88	80	76

The tabular data given in Tab. 1 shows that CNN, a deep learning algorithm, has achieved high accuracy rates, above 95 percent, for the second stage of classification. The other machine learning techniques have lesser accuracy scores in comparison with CNN. The accuracy achieved by Naïve Bayes is in the range of 61 to 83 percent in the second stage of the classification; the Decision tree achieves in the range of 70 to 83 percent; the Support Vector Machine in the range of 70 to 87 percent, and Random Forest in the range of 76 to 88 percent.

5 Conclusion and Future Scope

This paper evaluates the performance of various machine learning and deep learning approaches for Bug Classification of DevOps bugs using JIRA data. As opposed to the traditional method, in which a developer has to triage the bugs manually, this approach enables them to invest lesser time in debugging the issue and arriving at the root cause. Based on the experimental results, it can be observed that Convolutional Neural Networks, a deep learning technique, outperformed all the other approaches used. Furthermore, it was observed that Naïve Bayes, a probabilistic classifier generally preferred for text classification, performed poorly with the bug dataset used in this paper. Ensemble methods like a Decision tree and Random Forest performed better on this dataset. This model can be extended to suggest plausible solutions for the bugs, which can significantly reduce the maintenance windows. This approach can also be extended to support and resolve customer issues. Since the approach accepts inputs in textual format, this can also be extended for any text classification problem.

Acknowledgments: The authors thank their families and colleagues for their continued support.

Funding Statement: The author(s) received no specific funding for this study.

Availability of Data and Materials: The data used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Atlassian JIRA Software, "Bug tracking done right with Jira Software," 2020. [Online]. Available: https://www.atlassian.com/software/jira/bug-tracking

[2] Progress Chef, "Progress Chef is the Only DevOps Provider Recognized as a Policy as Code Leader," 2020. [Online]. Available: https://chef.io/

[3] D. Behl, S. Handa and A. Arora, "A Bug Mining Tool to Identify and Analyze Security Bugs using Naive Bayes and TF-IDF," *In. Proceedings of International Conference on Reliability Optimization and Information Technology*, Faridabad, India, pp. 294 – 299, 2014. https://doi.org/10.1109/ICROIT.2014.6798341

[4] P. Terdchanakul, H. Hata, P. Phannachitta and K. Matsumoto, "Bug or Not? Bug Report Classification using N-Gram IDF," *In. Proceedings of IEEE International Conference on Software Maintenance and Evolution* (ICSME), Shanghai, China, pp. 534 - 538, 2017. https://doi.org/10.1109/ICSME.2017.14

[5] T. Zhang and B. Lee, "A Bug Rule based Technique with Feedback for Classifying Bug Reports," *In. Proceedings of IEEE 11th International Conference on Computer and Information* Technology, Pafos, Cyprus, pp. 336 - 343, 2011. https://doi.org/10.1109/CIT.2011.90

[6] Neelofar, M. Y. Javed and H. Mohsin, "An Automated Approach for Software Bug Classification," *In. Proceedings of Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, Palermo, Italy, pp. 414 - 419, 2012. https://doi.org/10.1109/CISIS.2012.132

[7] C. Liu, J. Yang, L. Tan and M. Hafiz, "R2Fix: Automatically Generating Bug Fixes from Bug Reports," *In. Proceedings of IEEE Sixth International Conference on Software Testing, Verification and Validation*, Luxembourg, pp. 282 - 291, 2013. https://doi.org/10.1109/ICST.2013.24

[8] Linux Academy, "Learn by doing with A Cloud Guru," 2020. [Online]. Available: https:// https://acloudguru.com/

[9] Sous Chefs, "Sous Chefs are a community of Chef cookbook maintainers," 2020. [Online]. Available: https://github.com/sous-chefs

[10] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, Vol. 1, no. 1, pp. 81–106, 1986. https://doi.org/10.1007/BF00116251

[11] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1 pp. 5–32, 2001. https://doi.org/10.1023/A:1010933404324

[12] M. A. Hearst, S.T. Dumais, E. Osuna, J. Platt and B. Scholkopf "Support Vector Machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 1, pp. 18–28, 1998. https://doi.org/10.1109/5254.708428

[13] S. Xu, "Bayesian Naïve Bayes classifiers to text classification," *Journal of Information Science*, vol. 44, no. 1, pp. 48–59, 2018. https://doi.org/10.1177/0165551516677946

[14] Y. Luan and S. Lin, "Research on Text Classification Based on CNN and LSTM," *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Dalian, China, pp. 352-355, 2019. https://doi.org/10.1109/ICAICA.2019.8873454