DOI : https://doi.org/10.56452/7-4-244

ISSN: 0974-5823 Vol. 7 No. 4 April, 2022

International Journal of Mechanical Engineering

Detection of location of errors in code Using Deep Learning Ensembling methods

Arun Kumar Rai
Asst. Professor, Department of Computer Science, Graphic Era Hill University, Dehradun, Uttarakhand India 248002

Abstract:

Machine learning has the capacity of collecting raw data from a repository and converting it into a form that can be used for calculation and prediction of software bugs. It is usually preferable to identify a software problem as soon as possible in order to minimise the amount of time and money spent on fixing it. It is necessary to apply the wrapper and filter technique in order to discover the most optimum software metrics. The primary goal of this article is to identify the most effective model for predicting software bugs. The goal of certain machine learning applications is to learn characteristics of data sets in situations when the right solutions are not previously known by human users. Because there is no trustworthy test oracle for machine learning software, it is difficult to test such software.

Key words: Machine Learning, classification, clustering, Support Vector Machine, Software Defect Prediction.

1. Introduction

Machine learning methods are a crucial field of computer science that may be used to predict the development of software problems. They are becoming more popular. A software bug is one of the most severe issues that may occur in the computer industry and is one of the most difficult to detect. It is usually desirable to have the fewest amounts of software defects and for the software system to attain the greatest degree of accuracy feasible. This is especially true for mission-critical software systems. As an example of machine learning models, the regression technique is used in this article to model Linear Regression. Other machine learning models discussed in this article include the Random Forest model, the Neural Network model, the Support Vector Machine model, the Decision Tree model, and the Decision Stump model. More specifically, the goal of this research is to identify the most effective machine learning model that may be used to forecast software bug occurrences. In this article, we suggest a technique of software testing that is meant to address this problem in some way.

As software's dependence on and complexity grows, the need for maintainable, high-quality, low-cost software grows in parallel with these advancements as well. A software defect prediction system, on the other hand, is needed in order to reduce the amount of maintenance required in software operation while simultaneously improving the quality of software in

real- world situations. Putting in place an early detection system will enable developers to provide maintainable software in a shorter period of time since it will allow for the rapid correction of any issues that are discovered. There has been a significant increase in interest in the field of software engineering, particularly in the area of software defect prediction, during the last 30 years. Different performance comparison metrics, such as correlation, R Squared, mean square error (MSE), and accuracy, were calculated for a variety of machine learning models, and the results were then compared to one another. As for the rest of the paper, it was organised in the following way: It is the purpose of this article to discuss how to ensure the dependability of machine learning (ML) systems, with a special focus on software testing. In particular, traditional software engineering processes and tools do not easily translate to machine learning applicationsIt is a general phrase that is used to describe a wide group of software systems thatdo not have access to a trustworthy test oracle that can be used to test them. It is possible to show theoretically that ML algorithms of the highest quality exist, but this does not guarantee that an application will correctly implement or use the technique; as a result, software testing is needed [2].

The three elements of our approach for producing test cases are as follows: analysis of the problem domain and related data sets; analysis of the algorithm as stated; and analysis of implementation runtime decisions during the implementation phase.

2. Machine Learning Models

Machine learning is a subfield of Artificial Intelligence that is concerned with the development of a system that will learn from and make predictions based on data collected. As everycompany and individual tries to develop an intelligent application, machine learning has climbedto the top of the list of the most popular subjects on the internet. Unsupervised learning and supervised learning are the two kinds of machine learning that may be carried out at the same time. When data is analysed without supervision, unsupervised learning may be used to find hidden patterns in it. Clustering is a technique for learning without the need of a supervisor. The technique of supervised learning is used when it is required to train a model in order to make a prediction. This research was able to predict the optimum model for software bug prediction using regression methods applied to machine learning models, which was subsequently utilised in a later paper. This research made use of six different machine learning models [6].

A. Linear Regression

Predictive analysis is often carried out with the use of linear regression, which is a statistical method. To identify the relationship between the response variable (dependent variable) and one or more of the explanatory factors (independent variables), this model is employed (independent variable).

B. Decision Tree

If you're looking for predictive modelling methods, the decision tree is one of the supervised learning techniques that may be used in classification and regression. As an added benefit of

using decision trees, they are capable of coping with datasets that include errors and missing information. On the downside, decision trees are too sensitive to the data in the training set, which may include irrelevant or noisy information.

C. Support Vector Machine

Vector machine learning makes use of supervised computer learning methods such as classification, regression, and outliers detection to assist with the learning process. SVM works well when data sets are small, since the amount of time required for training is decreased, resulting in faster results. As data sets become less noisy, they become more appropriate models for the problems they are presented with. Statistical learning machines (SVMs) are used in a number of applications, including face identification, optical character recognition (OCR), spam categorization, financial time series forecasting, and other areas.

D. Neural Network

In order to establish the connection between input and output, to predict software problems and to detect patterns, it is feasible to use neural networks.

E. Decision Stump

Choice Stump is a machine learning model that mimics the process of making a decision in a given situation. The Decision Stump may be conceived of as a one-level decision tree with a single decision point. When the feature selected has significant values, the choice process generally results in the best outcome or results in the decision process continuing to develop.

3. Background

The creation of an artificial intelligence (AI) application that contains dynamic information about the present gadgets is something we're working on as a group. Because every device will ultimately fail, ("will fail" vs. "will not fail") is inadequate because, given enough time. While the prototype application generates rankings utilising both MartiRank and SVM algorithms, it is more than just a piece of academic research since it has real-world implications rather than beingsimply academic in nature.

3.1 Machine learning fundamentals

Supervisory machine learning is often split into two phases, which are as follows: Each example includes a number of attribute values as well as a single label, all of which are examined in the first phase of the investigation (known as the learning phase). After analysing a collection of training data that includes instances with multiple attribute values as well as a single label, the first phase (also known as the learning phase) is performed. In the end, this study produced a model that attempted to make broad generalisations about the connection between the qualities and the label. This is done in the second phase, when the model is applied to a fresh, previously unknown data set (the testing data), where the labels are not known beforehand (this is known as the validation data).

3.2 SVM and MartiRank

SVM is a machine learning method that belongs to the “linear classifier” family of algorithms. It is charged with finding a (linear) hyperplane that differentiates samples from different classes based on their characteristics. Because each example from the training data includes K characteristics, the SVM treats each example from the training data as a vector with K dimensions (and attempts to separate the instances using a hyperplane with K-1 dimensions) during the learning phase. The SVM's "kernel" influences the kind of hyperplane that is generated; in this article, we look at the linear, polynomial, and radial basis kernels, among otherthings.

3.3 Maximum Voting Classifier (MVC)

As a result of using the maximum voting technique, a large number of classifiers may be used tomake predictions and evaluate the data that they produce in real time. The final prediction is based on which candidate got the most number of votes in the previous election (more thanhalf). If you want to enhance the accuracy of this technique, you may combine multiple classifiers together.

3.4 Extra Tree Classifier (ET)

The numerical input figures used in this technique are used to further randomise a tree construction. The technique would transition. There is no rhyme or reason to the method by which the cut of point is selected. In particular, this approach is ideally suited to situations where there are a high number of numerical features that are subject to fluctuations. The method, which employs smoothing to enhance accuracy while also minimising any computational problems associated with finding the location of random forests and conventional trees, has been shown tobe successful.

3.5 Passive Agressive Classifier (PAC)

If the model falls into the proper kind of categorization classification, it is kept under the Passiveclassification. All incorrect classifications should be rectified in aggressive mode in order to account for the misclassified case that was shown. While it is true that updates are hampered bya lack of sufficient information, it is also true that a better model will aid in the correction of anymistakes that occurred during the previous time period.

3.6 Xgboost

A gradient-boosted decision tree method created by the Distributed Machine Learning Community is known as Gboost (DMLC). When used in combination with gradient-boosted decision trees, it is well-known for delivering increased performance and speed as a result. Tianqi Chen developed and used Xgboost for the first time in the year (1995), and it has shownto be a very successful way of giving a boost to computers since its inception. This project has gone through a number of revisions under the supervision of many different developers. Methods such as treestreme and xGBoost are used to help tree boosting approaches in making full use of all available hardware and memory resources. This allows

for their deployment in computing settings, as well as for the modification and improvement of the algorithm. Gradient boosting may be accomplished in XGBoost via the use of three distinct techniques: stochastic boosting, regularised boosting, and gradient boosting, among others (see below). Additionally, it has been very effective in modifying and adding regularisation parameters, making the best use of available memory resources, and reducing the amount of time spent performing computational operations on the data. XGBoost may also be used to perform operations on data that has already been given to the trained model, according to the documentation. It allows for the use of parallel structures and automatically fills in any missing data (Sparse Aware).

3.7 Gaussian Naïve Bayes (GNB)

When applied to both multi-class and binary (two-class) classification issues, this classification method, which may be stated using category or binary input values, can be very easy to comprehend. When used to multi-class classification issues, it may also be very easy to comprehend and implement. A variation of the Gaussian Naive Bayes model, commonly known as the Gaussian Naive Bayes model, is a model that allows for the extension of Naive Bayes to include actual value features.

4 METHODOLOGY

The method of research is also one of the most important components of any system's success in attaining its goal of obtaining success. The main aim of this article is to determine which machine learning model is the best successful at forecasting the incidence of software bugs. To accomplish this objective, a methodical methodology is necessary, as is the development of a framework for software defect prediction that makes use of previous data sets.

A. Data Collection

Data collection is also one of the most essential aspects of a system to pay close attention to during the development process. This experiment and analysis are based on data that is collected from the open source Promise repository, which has been validated and made publiclyavailable by the project's creators. A wide range of software metrics, such as product metrics and process metrics, are easily available for the purpose of software bug prediction and forecasting. During the trial, a product metrics suite called the Chidamber and Kemerer object- oriented (CK OO) metrics suite (which is a subset of object-oriented metrics) was utilised as theobject-oriented metrics suite. Illustration of the software components that were used in the research shows in Figure 1.

DOI : https://doi.org/10.56452/7-4-244

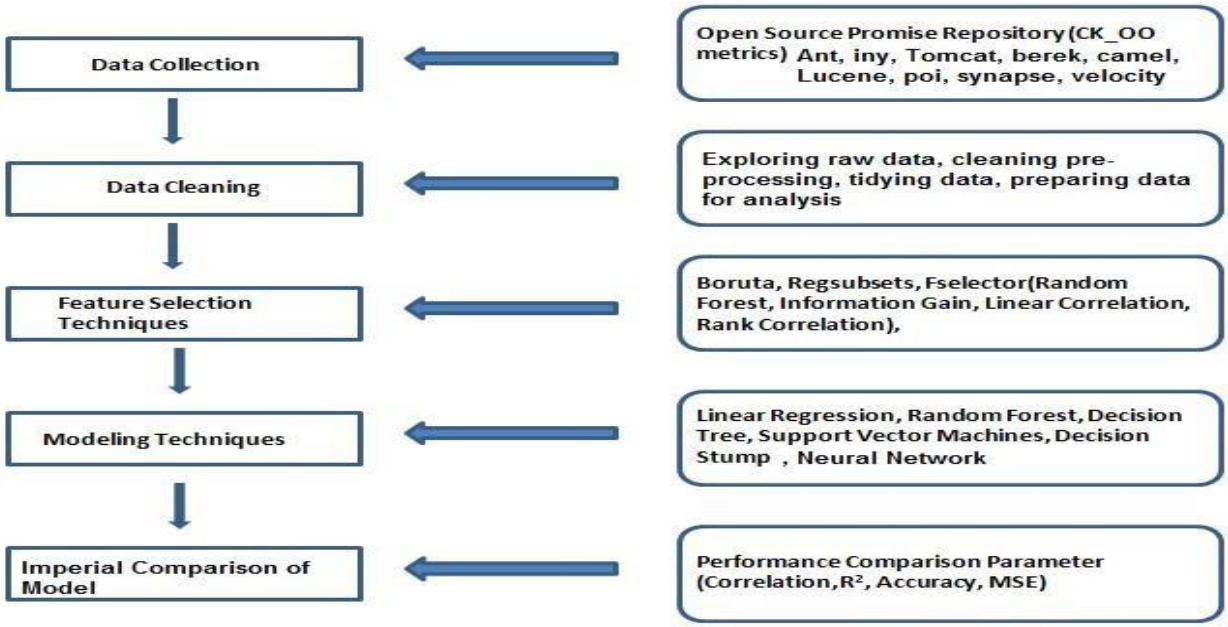

Fig.1. Framework for Software Defect prediction using Historical Databases

B. Data Cleaning

It is helpful in improving the quality of data since it handles with a variety of issues such as detecting discrepancies, removing errors, and restoring missing information. There is an issue with data cleansing in a data warehouse, which is a key component of the extraction, transformation, and loading (ETL) process. There are many different types of data cleaning technologies available, but occasionally a substantial portion of the data must be cleaned manually, which is time-consuming and difficult to produce and manage in the long run.

C. Feature Selection

The selection of features for bug prediction is an extremely essential stage since it affects the accuracy of the forecast as well as the complexity of the model. The use of the Feature Selection technique raises another significant point to consider: if the number of variables is more than what is considered optimal, accuracy of the machine learning algorithm will deteriorate as a consequence of the increased complexity of the algorithm. As a consequence, inorder to get the best possible outcomes, it is recommended to utilise a restricted feature set wherever possible.

D. Mathematical Model

Machine learning technique regression may be used to create a prediction model, which can then be used to make predictions. It is one of the methods that are accessible. Machine learning models' performance was evaluated in this study using the performance metrics.

DOI : https://doi.org/10.56452/7-4-244

5. Data Preparation

In order to attain software reusability, maintainability, and quality, it is necessary to use machine-learning techniques. This is because machine-learning methods assist in the identification of foul odours, ambiguities, errors, and defects inside software. The identification of software, on the other hand, may be achieved via the use of machine learning techniques.

Using pre-processing, you may assist in the translation of data into a format that can be read and utilised by the classification engine. There are a number of significant advantages to pre-processing data, including the ability to normalise numeric data and assist with filling in missing data. In order to conduct the tests, it was necessary to rely on datasets obtained from the PROMISE data repository, which had been gathered from actual NASA software projects and included a variety of software modules, as well as datasets obtained from other sources. The utilisation of publicly accessible datasets was required for the purposes of the benchmarking procedure. Other researchers may use this technique of benchmarking to compare their results with those of other researchers in the field. The NASA MDP data sets were usedto generate the performance evaluator matrices used in this research. The binary character of the target variables in these data sets is shown by the fact that they have values of 1 indicating yes and 0 indicating no. The Python programming language and the Scikit-learn library are used in the data inspection procedure (a machine learning framework).

Table 1: Description of NASA MDP DATSETS

Variables	Description	Metrics Type
loc	Line count of Code	McCabe
v(g)	Cyclomatic Complexity	McCabe
ev(g)	Essential Complexity	McCabe
iv(g)	Design Complexity	Halstead
n	Total operators and Operands	Halstead
v	Volume	Halstead
l	Program Length	Halstead
d	Difficulty	Halstead
i	Intelligence	Halstead

6. Software Testing Using Machine Learning Techniques

6.1 Testing SVM-Light

We were unable to find any issues with SVM-Light in the majority of the test scenarios that included unexpected To the best of our knowledge, none of our testing has shown that this is the case for the radial basis kernel in question. A machine learning researcher who is acquainted with SVM-Light, on the other hand, claims that the implementation uses "chunking," which implies that the optimization method operates on sections of data instead than the whole data set. In order to obtain fast convergence to the optimum, numerical methods and heuristics are used. However, the optimum state is not always achieved; rather, after achieving a given degree of development, the process grinds to a stop. For example, consider the following example of a critical area in which the implementation deviates substantially from the specification: our investigation of "predictable" rankings produced a number of other and noteworthy findings. A small data set wascreated with the assistance of the SVM that should provide a flawless ranking. All three kernels gave appropriate ratings to the samples. When the labels were changed in a subsequent test to make them all distinct (i.e., just equal to the first characteristic plus one), the findings were much more intriguing. When it came to determining the optimal ranking, the linear and radial basis kernels were successful, but the polynomial basis kernel was unsuccessful. The noise, we reasoned, was to blame, so we removed it, and it did, in fact, arrive at the optimum ranking position. This was the only case in which noise in the data set precluded SVM-Light from finding the most optimum ranking method for the given data set.

6.2 Testing MartiRank

When confronted with a larger than expected number of features, the implementation, in contrast to the MartiRank implementation, repeatedly failed to meet the requirements (crashed). On the other hand, negative labels were not detected by the implementation, which instead produced erroneous results whenever a negative label was encountered. According to theory, negative labels (such as -1 vs +1, which are commonly used in other applications) should be permitted by default in a general-purpose ranking system.

As an added bonus, we were able to construct test scenarios that showed that alternative interpretations of the algorithm might result in different results by carefully analysing the algorithm and taking into account any potential ambiguity. We were particularly interested in what would happen in the event of repeated values since MartiRank is based on sorting. We were particularly interested in whether “stable” sorting would be employed, which would guarantee. When we used non-stable sorting, we were unable to achieve perfect ranking because some ofthe examples could be out of order. We were able to obtain perfect ranking using a non-stable sort, but not certainly perfect ranking due to the possibility that the instances were out of order in the first place. Upon further investigation, we discovered that the sorting process was not, in fact,consistent.

DOI : https://doi.org/10.56452/7-4-244

6.3 Regression testing

The results of our tests enabled us to create an array of data sets, which we may then use for regression testing purposes in the future. This was a desirable side effect of our tests, and wewere happy to have it. The implementation of the MartiRank method is still in the early stages of development, but our data sets have shown to be very useful in identifying newly found faults in the system. A good example is the restructuring of some repetitious code and the placement of that code into a new subroutine, which showed that the models generated by the new code were different from those produced by the previous version of code.

7. Results

This section covers the outcomes of the various machine learning methods for defect prediction when applied to a variety of datasets. The training was carried out on the basis of a 10-fold cross validation.

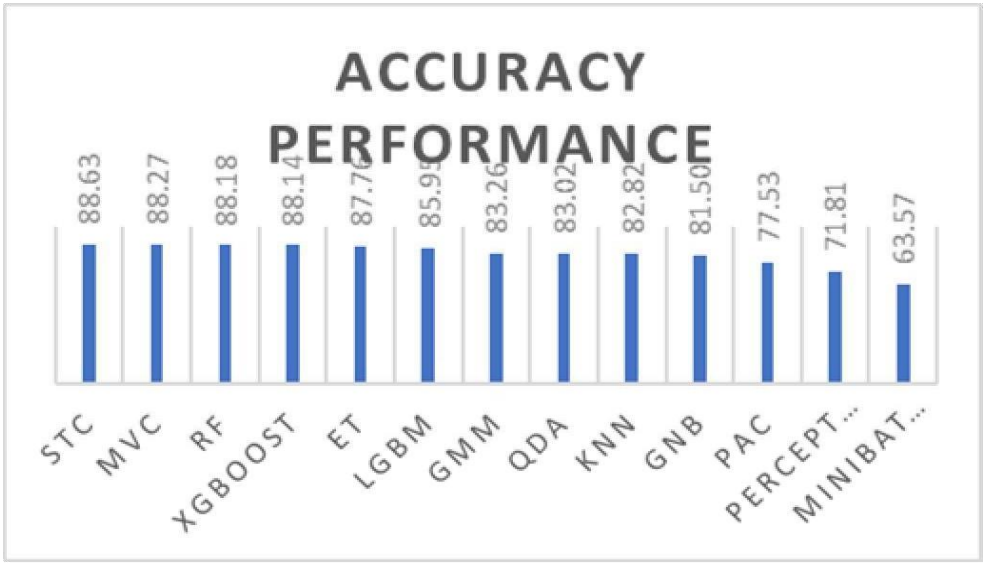

Fig 2: Accuracy Chart of Different algorithms

According to the Accuracy chart (Figure 2), it is immediately apparent that the proposed stacking classifier (STC) outperformed all of the other algorithms evaluated in this study significantly. The ensemble classifiers outperformed other supervised and unsupervised learning techniques in terms of accuracy when evaluated in terms of accuracy. In terms of classification technique performance, QDA beat the competition, but in terms of clustering algorithm performance, GMM topped the competition.

Conclusion

A rise in the number of software-based systems has occurred in recent years despite the fact that they must be thoroughly tested prior to being provided to end users. Software quality measures include ISO standards, CMM, and software testing, all of which may be used to enhance the overall quality of the product. Every passing day sees a rise in the need for

software testing, and software defect prediction may be able to assist improve the efficiency of software testing by predicting potential defects. A specific aim of this study was to investigate different software defect prediction techniques, which were recognised as the ensemble model, clustering models (for clustering models), and classification models (for classification models). The findings of this study show that stacking multiple classifiers may be used to predict faults in a machine learning system.

Reference:

1. J. Tian, and M.V. Zelkowitz. Complexity measure evaluation and selection, IEEE Transactions on SoftwareEngineering, 21(8), 641-650, 1995. https://doi.org/10.1109/32.403788
2. S. Puranik, P. Deshpande, and K. Chandrasekaran, "A Novel Machine Learning Approachfor Bug Prediction," Procedia - Procedia Comput. Sci., vol. 93, no.September, pp. 924–930, 2016. "doi:10.1016/j.procs.2016.07.271".
3. R.B. Jadhav, S.D. Joshi, U.G. Thorat, and A.S. Joshi. A Software Defect Learning and Analysis Utilizing Regression Method for Quality Software Development, International Journal of Advanced Trends in Computer Science and Engineering, 8(4), 1275 - 1282, 2019. https://doi.org/10.30534/ijatcse/2019/38842019
4. K.S. Kavya, and Y. Prasanth. An Ensemble DeepBoost Classifier for Software DefectPrediction, International Journal of Advanced Trends in Computer Science and Engineering, 9(2), 2021 – 2028, 2020. https://doi.org/10.30534/ijatcse/2020/173922020
5. M. Dhiauddin, M. Suffian, and S. Ibrahim, "A Prediction Model for System Testing Defects using Regression Analysis," Int. J. Soft Comput. Softw. Eng., vol. 2, no. 7,

pp. 55–68, 2012. "doi:10.7321/jscse.v2.n7.6"
6. M.K. Albzeirat, M.I. Hussain, R. Ahmad, F.M. Al-Saraireh, and I. Ahmad. A novel mathematical logic for improvement using lean manufacturing practices. Journal of Advanced Manufacturing Systems, 17(03), 391-413, 2018.
7. S. Aleem, L.F. Capretz, and F. Ahmed. Benchmarking machine learning technologies for software defect detection. International Journal of Software Engineering & Applications (IJSEA), 6(3), pp. 11-23, 2015.
8. E. Erturk, and E.A. Sezer. A comparison of some soft computing methods for software fault prediction. Expert systems with applications, 42(4), 1872-1879, 2015. https://doi.org/10.1016/j.eswa.2014.10.025
9. P. A. Selvaraj and P. Thangaraj, "Support Vector Machine for Software Defect Prediction," Int. J. Eng. Technol. Res., vol. 1, no. 2, pp. 68–76, 2013.
10. M.K. Albzeirat, M.I. Hussain, R. Ahmad, F.M. Al-Saraireh, A. Salahuddin, and N. Bin- Abdun. Applications of Nano-Fluid in Nuclear Power Plants within a Future Vision. International Journal of Applied Engineering Research, 13(7), 5528-5533, 2018.
11. F. E. H. Tay and C. Lijuan, "Application of Support Vector Machines in Financial Time Series Forecasting," Omega, vol. 29, no. 2001, pp. 309–317, 2001. "doi: 10.1016/s0305-0483(01)00026-3"
12. M. Singh, and D.S. Salaria. Software defect prediction tool based on neural network. International Journal of Computer Applications, 70(22), 2013.