**A Project Report**

*on*

**Software Fault Detection using Machine Learning**

*carried out as part of the* ***Minor Project IT3270*** *Submitted*

by

**Harsh Raj**

**219302137**

*in partial fulfilment for the award of the degree* *of*

**Bachelor of Technology**

in

**Information Technology**

Under the Guidance of

**Guide Name**

A close up of a sign

Description automatically generated

**School of Information, Security and Data Science**

**Department of Information Technology**

**MANIPAL UNIVERSITY JAIPUR**

**RAJASTHAN, INDIA**

**May 2024**

**CERTIFICATE**

Date:15/05/2024

This is to certify that the minor project titled **Software Fault Detection using Machine Learning** is a record of the bonafide work done by **Harsh Raj** (219302137) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Information Technologyof Manipal University Jaipur, during the academic year 2023-24.

**Dr Debolina Ghosh**

*Assistant Professor*

*Project Guide, Department of Information Technology*

*Manipal University Jaipur*

**Dr. Pankaj Vyas**

*HoD, Department of Information Technology*

*Manipal University Jaipur*

**ABSTRACT**

The growing complexity of software systems poses significant challenges to error prevention, underscoring the importance of automatically predicting errors in software modules. This enables developers to optimize their use of limited resources. Concerns about anticipating software bugs and vulnerabilities have long been prevalent among software developers and the technology industry. Traditionally, identifying software errors required expertise or a deep understanding of the application's problematic modules.

In this paper, I have introduced a machine learning-based predictive model for software defect evolution, designed to facilitate uninterrupted software operation. I have also evaluated the model's efficacy using established criteria such as accuracy, precision, recall, specificity, and F1 measures. Our findings demonstrate outstanding classification performance, ranging from 98% to 100%, achieved using SVM across three error datasets, as assessed by the F1 score. This study offers valuable insights for software researchers and practitioners, equipping them with automated decision-making capabilities to select appropriate tasks for their specific applications.

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Title** | **Page No** |
|  |  |  |
| 1 | Dataset details | 03 |
| 2 | Classification and performance of machine learning models | 05 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Title** | **Page No** |
|  |  |  |
| 1 | Architecture of the defect model | 03 |
| 2 | Representation of performances of machine learning models | 06 |
| 3 | Project Progress Chart | 06 |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| S.No |  | Page No. |
|  | Introduction  1.1 Overview  1.2 Motivation | 01 |
|  | Background Details  2.1 Literature Review | 02 |
|  | System Design and Methodology  3.1 System Architecture  3.2 Methodologies | 03 - 04 |
|  | Implementations and Results  4.1 Implementations  4.2 Results and Discussions | 04 - 06 |
|  | Conclusion and Future Work | 07 |
|  | Reference | 07 - 10 |

1. **INTRODUCTION**

1.1 *Overview-*

Throughout the software life cycle, defects cause time and cost overruns and provide serious difficulties to software development. Early defect detection lowers development costs while improving system quality and dependability. Studies have repeatedly demonstrated how effective software metrics are in predicting faults [15]. This paper focuses on evaluating popular machine learning techniques like—Logistic Regression, K-nearest Neighbors, Decision Tree, Random Forest, Naïve Bayes, and Support Vector Machine. The paper's emphasis on evaluating machine learning techniques aligns with the contemporary need for advanced approaches in fault prediction. As defects remain a pervasive challenge in software development, the application of machine learning models becomes paramount in identifying and addressing potential issues early on. By honing in on Logistic Regression, K nearest Neighbors, Decision Tree, Random Forest, and Support Vector Machine, the paper contributes to the ongoing discourse on selecting robust methodologies for fault prediction. This evaluation not only holds promise for optimizing the software development life cycle but also underscores the significance of leveraging advanced analytical tools to enhance the overall efficiency and effectiveness of fault prediction strategies.

The purpose of this research is to explore how well five different classifiers perform in automating the resolution of software bugs. Three datasets sourced from PROMISE (KC1, JM1, and CM1) [33] were employed for this investigation. To address data imbalance concerns, various computational methods including PCA, Resample, and SMOTE [32] were employed to allocate correlated columns to our pre-processed data. Consequently, the paper provides a comparative analysis of five machine-learning methods for estimating software faults.

1.2 *Motivation-*

Undertaking a project on software fault prediction using machine learning is motivated by the compelling prospects of reducing development costs and enhancing software quality. Early defect detection not only reduces debugging costs but also helps projects be completed on schedule and succeed by averting delays and enhancing system dependability. The project addresses the increasing complexity of software systems in line with industry trends and provides participants with an excellent learning opportunity that enhances their skills in software engineering and machine learning. Furthermore, the application of machine learning models facilitates the effective distribution of resources, enabling developers to concentrate on regions of high risk and maximize their endeavors throughout the software development life cycle.

1. **BACKGROUND DETAILS**

2.1 *Literature Review-*

Several studies have been conducted to improve software defect prediction accuracy, each employing different methodologies, approaches, tools, and techniques. Mehmood et al. (2023) utilized a two-tail t-testing approach and machine learning algorithms such as SVM, Random Forrest, Logistic Regression, and Bayes Net, using the WEKA tool. Their approach focused on early identification, dataset-centric strategies, and feature selection. However, their future work lacked specificity, and they did not adequately explore the software systems' complexity.

* Sushant Kumar Pandey et al. (2021) [30] conducted a survey on machine learning-based methods for software fault detection, employing various algorithms such as SVM, Logistic Regression, Bayes Net, ANN, and Decision Tree. Their methodical approach to research queries and score-based evaluation provided valuable insights, but they didn't discuss the generalizability of their results or the limited availability of datasets.
* Qiao et al. (2019)[31] proposed a deep learning-based software defect prediction method utilizing SVM, FSVR, and DTR algorithms. They aimed for versatility and change-level predictions but acknowledged the risk of overfitting and data availability hurdles.
* Xing et al. (2019)[18] focused on fault detection for web services using Naïve Bayes classification, emphasizing automation and utilization of service execution logs. However, they acknowledged limitations such as the Naïve Bayes assumption and threshold sensitivity.
* Phuong Ha et al. (2019) [3] aimed to improve software defect prediction accuracy through an experimental study, employing various algorithms like Logistic Regression, SVM, Decision Tree, MLP, KNN, Naïve Bayes, and Random Forrest. Their study highlighted focused metric usage, performance comparisons, and practical applicability but lacked exploration of classification techniques and metrics selection rationale.
* Rudenko et al. (2018)[21] proposed a software faults number evaluation method based on correction of experimental data exponential line using statistical tools like Standard Deviation and X^2 Test. Their method emphasized simplicity and technical ease but made assumptions about linearity and faced challenges with data quality and fault interaction complexity.
* Alakus et al. (2019)[25] utilized a quality matrix for estimating software faults, emphasizing client involvement, risk management, and quality assurance. However, their study had limited metric coverage, and generalization challenges, and did not address data collection challenges.
* Cetiner and Sahingoz (2020)[29] analyzed machine learning-based software defect prediction systems using algorithms like SVM, Decision Tree, KNN, Naïve Bayes, and Random Forrest. Their study emphasized the validation of proposed models and enhanced software quality but faced limitations in dataset selection and the risk of overfitting.

1. **SYSTEM DESIGN AND METHODOLOGY**

3.1 *System Architecture-*

*A diagram of data processing

Description automatically generated*

**Figure 1:** Architecture of the defect model

3.2 *Methodologies*

Data Collection: 3 open sources of freely accessible data from the PROMISE Software Engineering Database are used in this experiment. The metrics used are categorized based on their types, such as Halstead and McCabe metrics[36], as well as miscellaneous ones. McCabe metrics include line of code, cyclomatic complexity, essential complexity, and design complexity, which aid in assessing code complexity and structure. Meanwhile, Halstead metrics, such as operators and operands, volume, program length, line count, intelligence, Blank line counts, comments count, and effort[36], offer insights into program complexity, size, and effort required for development[36]. Additionally, arbitrary metrics such as input/output code and comments, unique administrators and operands, sum of administrators and operands, and department checks provide advanced configuration on program features. The presence or absence of defects is also recorded as a binary metric. Collectively, these metrics plays an essential role in software quality assessment and defect prediction, aiding developers and researchers in understanding, evaluating, and improving software systems.

**Table 1:** Dataset details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Name | Total | True | False |
| 1 | KC1 | 2109 | 326 | 1723 |
| 2 | CM1 | 498 | 49 | 449 |
| 3 | JM1 | 10885 | 8779 | 2106 |
|  |  |  |  |  |

**Measurement Criteria:**

Accuracy = (𝑇N + 𝑇P)

(F𝑃 + 𝑇𝑁 + TP + 𝐹𝑁)

Recall = TP

(FN + TP)

Precision = 𝑇𝑃

(F𝑃 + T𝑃)

Specificity = 𝑇𝑁

(𝐹𝑃 + TN)

F1 measure = (𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 \* Recall)\*2

(𝑅𝑒𝑐𝑎𝑙𝑙 + 𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛)

1. **IMPLEMENTATION AND RESULTS**

4.1 *Implementation Details-*

Implement six classification methods: Decision Tree (DT), Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), Logistic Regression (LR). Employ 10-fold cross-validation for robust evaluation. Assess model performance using metrics including Accuracy, Recall, F1 Score, and Precision. Preprocess datasets to enhance accuracy and consistency. Divide datasets into 80% training and 20% testing sets. Train classifiers on the training data and evaluate on testing data.

4.2 *Results and Discussion-*

The consideration centered on robotized blame recuperation inside the program, utilizing a proactive demonstration. Different stages were watched, counting Prerequisite Detail, Necessity Investigation, Physical Plan, HLD (Tall Level Graph), DLD (Detail Level Graph), Sending, Program Imperfection Expectation, Advancement, Testing, Consistent Plan, Begin, Halt, Information Pre-processing, applying 5 classifiers, Analysing Demonstrate Execution, taking care of 3 Imperfection Datasets, and part the datasets (Preparing 80% & Testing 20%). 10-fold cross-validation was used in the investigation to evaluate the performance of five classification processes. Diverse information preprocessing strategies were utilized to upgrade the exactness and consistency of the classification show. The execution assessment of the five administered classification strategies for program blame expectation uncovered.

**Accuracy:**

DT and SVM accomplished 99% on JM1 datasets Random Forest, SVM, and Decision Tree accomplished 100% on CM1 datasets; Decision Tree, SVM, and Random Forest got 100% on KC1 datasets.

**Recall:**

SVM and Random Forest performed best on JM1 datasets; LR appeared the least executed on CM1 and KC1 datasets.

**F1 degree:**

SVM accomplished 100% on JM1 datasets. For CM1 datasets, the F1 scores were generally comparative among classifiers ( Random Forest, SVM, Decision Tree = 100%, KNN = 97%, Logistic Regression = 95%). Random Forest accomplished the most elevated score (99%), and Logistic Regression was the most reduced (89%) on KC1 datasets.

**Precision:**

All classifiers illustrated amazing execution on JM1, CM1, and KC1 datasets, showing adequacy in foreseeing computer program imperfection modules.

**Table 2:** Classification and performance of machine learning models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | |  |  |  | | --- | --- | --- | | KC1 | CM1 | JM1 | |
| KNN | |  | | --- | | Accuracy | | Precision | | Recall | | F1 | | |  |  |  | | --- | --- | --- | | .99 | .99 | .99 | | .89 | .95 | .95 | | .96 | 1.0 | .97 | | .92 | .97 | .95 | |
| DT | |  | | --- | | Accuracy | | Precision | | Recall | | F1 | | |  |  |  | | --- | --- | --- | | 1.0 | 1.0 | .99 | | 1.0 | 1.0 | 1.0 | | 1.0 | 1.0 | .99 | | 1.0 | 1.0 | .99 | |
| Logistic Regression | |  | | --- | | Accuracy | | Precision | | Recall | | F1 | | |  |  |  | | --- | --- | --- | | .98 | .98 | .97 | | 1.0 | .95 | .94 | | .80 | .95 | .92 | | .89 | .95 | .94 | |
| SVM | |  | | --- | | Accuracy | | Precision | | Recall | | F1 | | |  |  |  | | --- | --- | --- | | 1.0 | 1.0 | .99 | | 1.0 | 1.0 | 1.0 | | 1.0 | 1.0 | 1.0 | | 1.0 | 1.0 | 1.0 | |
| Random Forrest | |  | | --- | | Accuracy | | Precision | | Recall | | F1 | | |  |  |  | | --- | --- | --- | | 1.0 | 1.0 | .99 | | 1.0 | 1.0 | .99 | | 1.0 | 1.0 | 1.0 | | 1..0 | 1.0 | .99 | |

A graph of multiple colored lines

Description automatically generated with medium confidence

**Figure 2:** Representation of performances of machine learning models

4.3 *Progress Chart*

**A graph with colorful bars

Description automatically generated with medium confidence**

**Figure 3:** Project Progress Chart

1. **CONCLUSION AND FUTURE WORK**

In this paper, I have advertised a robotized program designing the approach to create imperfection forecast (SDPD) models amid the program advancement lifecycle. At this point, the primitive objective of my experiment was to 'estimate the capacity of five-based administered machines to memorize classification methods to anticipate computer program bug modules utilizing 3 NASA datasets.

The comes about (i.e. Precision: 89-100%) of testing with distinctive properties illustrate the capacity and viability of the SDPD show in distinguishing blunders and making strides in program quality. Furthermore, this SDPD show that can identify program bugs early by collecting real-time computer program advancement information from target applications. The offered method can be utilized for computer program crash recuperation in frameworks and improved by implementing machine learning methods to form a more productive SDPD in program crash recuperation.

In the future, I plan to perform additional classification calculations, such as combination or outfit models, to confirm program crash expectations.

**REFERENCES**

[1] X. Zhang, X. Chen, J. Wang, Z. Zhan, and J. Li, “Verifiable privacy-preserving single-layer perceptron training scheme in cloud computing,” Soft Computing, vol. 22, no. 23, pp. 7719–7732, 2018.

[2] B. Dhana Laxmi, Dr. G Apparao Naidu, Dr. K Anuradha, “A Review on Software Fault Detection and Prevention Mechanism in Software Development Activities,” IOSR Journal of Computer Engineering (IOSRJCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 17, Issue 6, PP. 25-30, Ver. V Nov – Dec. 2015.

[3] Le Hoang Son, Nakul Pritam, Manju Khari, Raghvendra Kumar, Pham Thi Minh Phuong, and Pham Huy Thong, “Empirical Study of Software Defect Prediction: A Systematic Mapping,” MDPI journals,2019.

[4] Pooja Sharma, Amrit Lal Sangal, “Examining the Predictive Capability of Advanced Software Fault Prediction Models – An Experimental Investigation Using Combination Metrics,” e-Informatica Software Engineering Journal, Volume 16, Issue 1, pages: 220104, DOI 10.37190/e-Inf220104, 2022.

[5] Z. Li, X.Y. Jing, and X. Zhu, “Progress on approaches to software defect prediction,” Iet Software, Vol. 12, No. 3, pp. 161–175. 2018.

[6] Danijel Radjenović, Marjan Heričko, Richard Torkar, Aleš Živkovič, Software fault prediction metrics: A systematic literature review, Information and Software Technology, Volume 55, Issue 8,2013, Pages 1397-1418, ISSN 0950 584, <https://doi.org/10.1016/j.infsof.2013.02.009>.

[7] Yasutaka Kamei, Emad Shihab., “Defect Prediction: Accomplishments and Future Challenges,” Yasutaka Kamei on 31 December 2017.

[8] Tianwen Yao , Ben Zhang, Jun Peng, Zhiqiang Han, Zhaobing Yang, Zhi Zhang, and Bo Zhang,” Defect Prediction Technology of Aerospace Software Based on Deep Neural Network and Process Measurement” Hindawi Mathematical Problems in Engineering Volume 2022, Article ID 1276830, 8 pages https://doi.org/10.1155/2022/1276830 , Published 1 February 2022.

[9] Muhammad Adnan Khan, Nouh Sabri Elmitwally, Sagheer Abbas, Shabib Aftab, Munir Ahmad, Muhammad Fayaz, and Faheem Khan” Software Defect Prediction Using Artificial Neural Networks: A Systematic Literature Review” Hindawi Scientific Programming, Article ID 2117339, 10 pages https://doi.org/10.1155/2022/2117339, Published 12 May 2022.

[10] Qiuying Li 1,2,\* and Hoang Pham,” Modeling Software Fault-Detection and Fault-Correction Processes by Considering the Dependencies between Fault Amounts”, https://www.mdpi.com/journal/applsci, Appl.Sci.,11,6998.https://doi.org/10.3390/app11156998, 2021.

[11] Jinyin Chen, Keke Hu, Yitao Yang, Yi Liu, Qi Xuan “ Collective transfer learning for defect prediction” www.elsevier.com/locate/neucom, 2019.

[12] Devika S, Lakshmi P L,” Best Suited Machine Learning Techniques for Software Fault Prediction” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020.

[13] Venkata U and R. A, "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques " Proceedings of the 10th IEEE International Workshop on Object-oriented Real-Time Dependable Systems 2005.

[14] <http://www.academia.edu/>

[15] Software Fault Prediction- A Roadmap By Sandeep Kumar & Santosh Singh Rathore.

[16] Arisholm, E. (2004). Dynamic coupling measurement for object-oriented software. IEEE Transactions on Software Engineering, 30(8), 491–506.

[17] Bansiya, J., & Davis, C. (2002). A hierarchical model for object-oriented design quality assessment. IEEE Transactions on Software Engineering, 28(1), 4–17.

[18] X. Xing, J. Luo, Z. Jia, Y. Li, and Q. Han, "Automated Fault Detection for Web Services using Naïve Bayes Approach," 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2019, pp. 336-339, doi: 10.1109/ICSESS47205.2019.9040756.

[19] Briand, L., Devanbu, P., & Melo, W. (1997). An investigation into coupling measures for C++. In Proceedings of the 19th International Conference on Software Engineering (pp. 412–421).

[20] Bundschuh, M., & Dekkers, C. (2008). The IT measurement compendium. Bunescu, R., Ruifang, G., Rohit, J. K., Marcotte, E. M., Mooney, R. J., Ramani, A. K., et al. (2005). Comparative experiments on learning information extractors for proteins and their interactions [special issue on summarization and information extraction from medical documents]. Artificial Intelligence in Medicine, 33(2), 139–155.

[21] O. Rudenko, E. Odarushchenko, Z. Rudenko and M. Rudenko, "The secondary software faults number evaluation based on correction of the experimental data exponential line approximation," 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kyiv, Ukraine, 2018.

[22] Chidamber, S., & Kemerer, C. (1994). A metrics suite for object-oriented design. IEEE Transactions on Software Engineering, 20(6), 476–493.

[23] Conte, S. D., Dunsmore, H. E., & Shen, V. Y. (1986). Software engineering metrics and models. Benjamin-Cummings Publishing Co., Inc.

[24] Crasso, M., Mateos, C., Zunino, A., Misra, S., & Polvorín, P. (2014). Assessing cognitive complexity in java-based object-oriented systems: Metrics and tool support. Computing and Informatics, 32.

[25] T. B. Alakus, R. Das and I. Turkoglu, "An Overview of Quality Metrics Used in Estimating Software Faults," 2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*,* Malatya, Turkey, 2019, pp. 1-6, doi: 10.1109/IDAP.2019.8875925.

[26] Drummond, C., & Holte, R. C. (2006). Cost curves: An improved method for visualizing classifier performance. In Proceedings of the Machine Learning Conference (pp. 95–130).

[27] Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction Gu, Z., Zhang, J., & Ji, Y. (2019). Topology Optimizing in FSO-based UAVs Relay Networks for Resilience Enhancement. Mobile Networks and Applications.

[28]Arimoto, Y., Presi, M., Guarino, V., D’Errico, A., Contestabile, G., Matsumoto, M., & Ciaramella, E. (2008). 320 Gbit/s (8×40 Gbit/s) double-pass terrestrial free-space optical link transparently connected to optical fiber lines. 2008 34th European Conference on Optical Communication.

[29] M. Cetiner and O. K. Sahingoz, "A Comparative Analysis for Machine Learning based Software Defect Prediction Systems*,"*2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*,* Kharagpur, India, 2020.

[30]Sushant Kumar Pandey, Ravi Bhushan Mishra, Anil Kumar Tripathi, Machine learning based methods for software fault prediction: A survey, Expert Systems with Applications.

[31]Lei Qiao, Xuesong Li, Qasim Umer, Ping Guo, Deep learning-based software defect prediction, Neurocomputing.

[32] R. Malhotra and S. Kamal, “An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data,” Neurocomputing, vol. 343, pp. 120–140, May 2019.

[33]“PROMISE DATASETS PAGE.” [Online]. Available:<http://promise.site.uottawa.ca/SERepository/datasets-page.html>

[34] Mehmood, Iqra, Sidra Shahid, Hameed Hussain, Inayat Khan, Shafiq Ahmad, Shahid Rahman, Najeeb Ullah, and Shamsul Huda. "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning." IEEE Xplore vol.20, pp. 23-28, 2023.

[35] R.A. Paul. "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques", 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2005.

[36] Mustafa Zaki Mohammed, Ibrahim Ahmed Saleh. "Predicted of Software Fault Based on Random Forest and K-Nearest Neighbor",2022 4th International Conference on Advanced Science and Engineering (ICOASE),2022