|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author (year)** | **Objective of the study** | **Methodology/approaches/ tools/techniques used** | **Advantages** | **Disadvantages** |
| Iqra Mehmood; Sidra Shahid; Hameed Hussain; Inayat Khan; Shafiq Ahmad; Shahid Rahman  (2023) | - To improve Software Defect Prediction Accuracy | * Two-tail t-testing * SVM * Random Forrest * Logistic Regression * Baye’s Net * WEKA tool used | **Early Identification**: Predicts flaws in source code before testing, enabling early defect identification.  **Dataset-Centric Approach**: Recognizes the dependency on dataset characteristics for choosing appropriate prediction methods.  **Feature Selection**: Utilizes feature selection, improving the accuracy of specific algorithms like Bayesian Net. | **Future Work is Broad and Generic**:  Future work is somewhat generic;  specific goals or hypotheses could  enhance its focus.  **Complexity of Software Systems**  **Not Explored**: Although the study mentions the increasing complexity of software systems, it does not delve into how this complexity may impact the accuracy and effectiveness of defect prediction models. |
| Sushant Kumar Pandey, Ravi Bhushan Mishra, Anil Kumar Tripathi  (2021) | - Survey of Machine  learning based  methods for  Software Fault  Detection | * SVM * Logistic Regression * Baye’s Net * ANN * Decision Tree | * **Research Queries and**   **Protocols**: The formulation of research queries and the development of a review protocol based on previous articles demonstrate a methodical and evidence-based approach to gathering information.  **Score-Based Evaluation**: The use of scores in Tables A.14 to A.16, along with corresponding categories in Table A.9, provides a quantitative assessment of the relevance of articles to SFP over ML. This scoring system helps in prioritizing and understanding the significance of each study. | **Generalization of Results**: While  the study indicates an average  AUC range and accuracy for ML-  based SFP models, it does not  explicitly discuss the  generalizability of these results to  diverse datasets or project types.  **Dataset Availability**: The call for  freely available datasets by  industries is mentioned, indicating  a limitation in the availability of  datasets for SFP research. This can  constrain the scope of experiments. |
| Lei Qiao, Xuesong Li, Qasim Umer, Ping Guo  (2019) | -Deep Learning  Based Software  Defect Prediction. | * SVM * FSVR * DTR | **Versatility**: The proposed approach aims to predict the number of defects in software modules, showcasing its applicability across different projects and datasets.  **Change-Level Predictions**: The plan to investigate defect predictions at the change level . | **Overfitting Risk:** While the paper reports improved performance, there is a potential risk of overfitting to the specific datasets used. It's essential to ensure that the model generalizes well to new and unseen data, especially when applied to different projects and programming languages.  **Data Availability Hurdles:** The reliance on new and commercial datasets may encounter challenges due to access restriction. |
| **Author (year)** | **Objective of the study** | **Methodology/approaches/ tools/techniques used** | **Advantages** | **Disadvantages** |
| Xing Xing, Jianyan Luo, Zhichun Jia, Yanyan Li, and Qiuyang Han(2019) | **-**Fault Detection for Web Services | **-** Naïve Bayes | **Automation:** The framework automates the fault detection process, reducing the need for manual intervention. This can save time and resources in identifying and addressing service faults**.**  **Utilization of Service Execution Logs:** Leveraging service execution logs for creating the training set is advantageous as it allows the model to learn from real-world data, capturing patterns and nuances specific to the actual service operation.  **Naïve Bayes Classification:** The Naïve Bayes approach is known for its simplicity and efficiency in classification tasks. It is particularly effective when dealing with a large number of features, making it suitable for processing service execution logs. | **Naïve Bayes Assumption:** The Naïve Bayes approach relies on the assumption of independence among features. In real-world scenarios, this assumption may not always hold, potentially leading to suboptimal performance in certain cases.  **Threshold Sensitivity:** The effectiveness of the model is contingent on an appropriately chosen threshold value. Setting this value too low may result in false positives, while setting it too high could lead to false negatives. Determining an optimal threshold may require additional tuning and experimentation.  **Limited to Log Data:** Depending solely on service execution logs might limit the model's ability to detect faults caused by issues outside the scope of logged events. It may miss faults that manifest in ways not captured by the available log data. |
| Thi Minh Phuong Ha, Duy Hung Tran, LE Thi My Hanh,  Nguyen Thanh Binh  (2019) | -To improve  Software Defect  Prediction Accuracy (experimental study) | * Logistic Regression * SVM * Decision Tree * MLP * KNN * Naïve Bayes * Random Forrest | **Focused Use of Object-Oriented Metrics:** The study leverages object-oriented metrics, providing a specific and relevant set of indicators for evaluating and improving software quality.  **Performance Comparison:** Results highlight that Support Vector Machine excels in predicting faults at the class-level, while Multilayer Perceptron stands out for method-level datasets. This provides valuable insights for practitioners in selecting appropriate techniques for different aspects of software development.  **Practical Applicability:** By using real-world PROMISE datasets, the study ensures the practical relevance of its findings, enhancing the applicability of the results in real- | **Limited Exploration of**  **Classification Techniques:** The  mention of future work involving the  study of classification techniques to  address dataset imbalance implies that  this aspect has not been fully explored  in the current research. This could  limit the depth of understanding  regarding dataset challenges.  **Metrics and Techniques Selection:**  The study doesn't elaborate on the  rationale behind the choice of specific  object-oriented metrics or why these  seven machine learning techniques  were selected. Providing this context  could enhance the credibility and  transferability of the findings.  **Potential Biases in Results:** The  focus on only seven machine learning  techniques may introduce biases, and  the performance ranking could be  influenced by the chosen set. A  broader exploration of various  techniques might provide a more  nuanced understanding. |
| Oleksander  Rudenko,  Elena  Odarushch  Enko,  Zinaida  Rudenko,  Maryna  Rudenko.  (2018) | -Software Faults Number  Evaluation Based  on Correction of  the Experimental Data Exponential Line | * Standard Deviation * X^2 Test | **Simplicity and Technical Ease:**  The proposed method is described as technically simpler compared to previous approaches that involve comparing the values of the trend of faults with corresponding values of a regression line. This simplicity can lead to easier implementation and application in practical scenarios.  **Consistency with Previous Results:**  The results obtained through this method are reported to be consistent with previously obtained results using a different approach. This consistency enhances the credibility and reliability of the proposed method, reinforcing its potential effectiveness.  **Efficiency in Estimation:**  By using statistical data and an exponential approximation, the method may offer efficiency in estimating the number of secondary faults. Exponential functions are known for their ability to capture growth trends, making them suitable for modeling fault propagation over time. | **Assumption of Linearity:**  The method assumes that faults can be approximated by a linear correction to an exponential trend. In real-world scenarios, fault propagation may involve nonlinear patterns, and this assumption might not hold universally.  **Dependence on Quality of Statistical Data:**  The accuracy of the estimation relies heavily on the quality of the statistical data. Inaccurate or incomplete data can lead to biased results and compromise the reliability of the estimated number of secondary faults.  **Complexity of Fault Interactions:**  If the method oversimplifies the interactions between primary and secondary faults, it may not adequately capture the complexity of fault propagation in intricate systems. This can result in underestimation or overestimation of secondary faults.  **Sensitivity to Parameter Tuning:**  Exponential approximations often involve parameters that need to be tuned. Sensitivity to the choice of these parameters could affect the accuracy of the estimation and may require careful calibration.  **Limited Insight into Root Causes:**  The method may provide estimates of secondary faults but may not offer insight into the root causes of these faults. Understanding the underlying reasons for fault propagation is crucial for effective system management and improvement. |
| **Author (year)** | **Objective of the study** | **Methodology/approaches/ tools/techniques used** | **Advantages** | **Disadvantages** |
| Talha Burak Alakus,  Resul Das,  Ibrahim Turkoglu  (2019) | **-**Quality Matrix Used in Estimating  Software Fault |  | **Client Involvement Emphasis:** The study emphasizes the positive effects of incorporating clients during software development, promoting robustness, reliability, security, and adaptability to unexpected requirements.  **Risk Management Importance:** The study underscores the importance of risk management in software projects, advocating for a well-developed risk management cycle that categorizes and estimates risks early in the project.  **Quality Assurance Recommendations:** Quality assurance is highlighted as a key factor for software products. Clear requirements, detailed planning, correct test scenarios, and early documentation are recommended to enhance the quality of the development process**.** | **Limited Metric Coverage:** The study focuses on only three metrics suites (CK, MOOD, and QMOOD), potentially excluding other relevant metrics that researchers might find useful for software fault prediction**.**  **Generalization Challenges:** The study provides recommendations for development approaches and risk management but may not consider the unique context of every software project. Generalizations may not be universally applicable.  **Data Collection Challenges:** While the study advises researchers on data collection, it does not delve into the challenges and potential biases associated with collecting and using proprietary datasets. |
| Murat Cetiner,  Ozgur Koray Sahingoz  (2020) | - Analysis for Machine Learning  based Software Defect Prediction Systems | * SVM * Decision Tree * KNN * Naïve Bayes * Random Forrest | **Validation of Proposed Models:** The study demonstrates that the proposed models have good accuracy in predicting software defects  **Enhanced Software Quality:** By employing software defect prediction models, software managers can identify potential errors | **Limited Dataset Selection:** The study focuses on a specific set of public datasets from the PROMISE warehouse. The limited dataset selection might affect the generalizability of the results to other software development contexts.  **Possible Overfitting:** The study demonstrates good accuracy rates, but there is a possibility of overfitting to the specific datasets used. The models may perform exceptionally well on these datasets but might not generalize as effectively to new, unseen data. |