A Minor Project Synopsis on

Software Fault prediction using Machine learning

Submitted to Manipal University Jaipur

towards the partial fulfillment for the award of the degree of

Bachelor of Technology

In Information Technology

By

Harsh Raj

219302137

IT VI B

Under the Guidance of

Dr. Debolina Ghosh

INSPIRED BY LIFE

MANIPAL UNIVERSITY
JAIPUR

Department of Information Technology

School of Information Technology Manipal

University Jaipur

2023-2024

Introduction

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. 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, Knearest Neighbors, Decision Tree, Random Forest, Naïve Bayes, 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.

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.

Project Objectives

Dataset Exploration and Preprocessing:

In dataset exploration, the dataset's nuances are investigated, addressing missing values, outliers, and feature characteristics. Preprocessing takes these insights to refine the dataset, handling data irregularities, scaling features for uniformity, and encoding variables for algorithm compatibility. This meticulous preparation ensures a robust dataset, setting the groundwork for machine learning models to discern meaningful patterns and contribute to accurate software fault predictions.

Identification of Effective Metrics:

This objective involves a meticulous analysis to identify the software metrics most influential in predicting faults accurately. Employing statistical methods and machine learning algorithms, the project assesses the relationships between diverse metrics and fault occurrence. Feature selection techniques, including Recursive Feature Elimination and tree-based algorithms, help pinpoint the metrics that significantly contribute to the predictive power of fault prediction models. The goal is to streamline the fault prediction process by focusing on a refined set of metrics, ensuring both quantitative accuracy and qualitative insights for developers in addressing potential software issues during development.

Early flaw detection in software:

Reported an investigation using requirement-level and design-level software metrics. They found out that the early-level software metrics used for fault prediction help in improving the software design and eventually lead to a lesser number of faults in the software system.

Testing cost reduction:

The effectiveness of test effort allocation strategies based on fault prediction results in terms of cost saving for a telecommunication system. The results of the study revealed that using fault prediction with testing achieved a 25% reduction in the testing efforts without affecting the faultfinding efficiency of the testing process.

Methodology

1 Project Definition and Scope:

a. Develop a machine learning model to detect and predict software faults, enhancing overall software reliability.

b. In-scope: Creation of a fault detection system using historical data.

c. Out-of-scope: Detailed fault remediation; focus is on detection, not resolution.

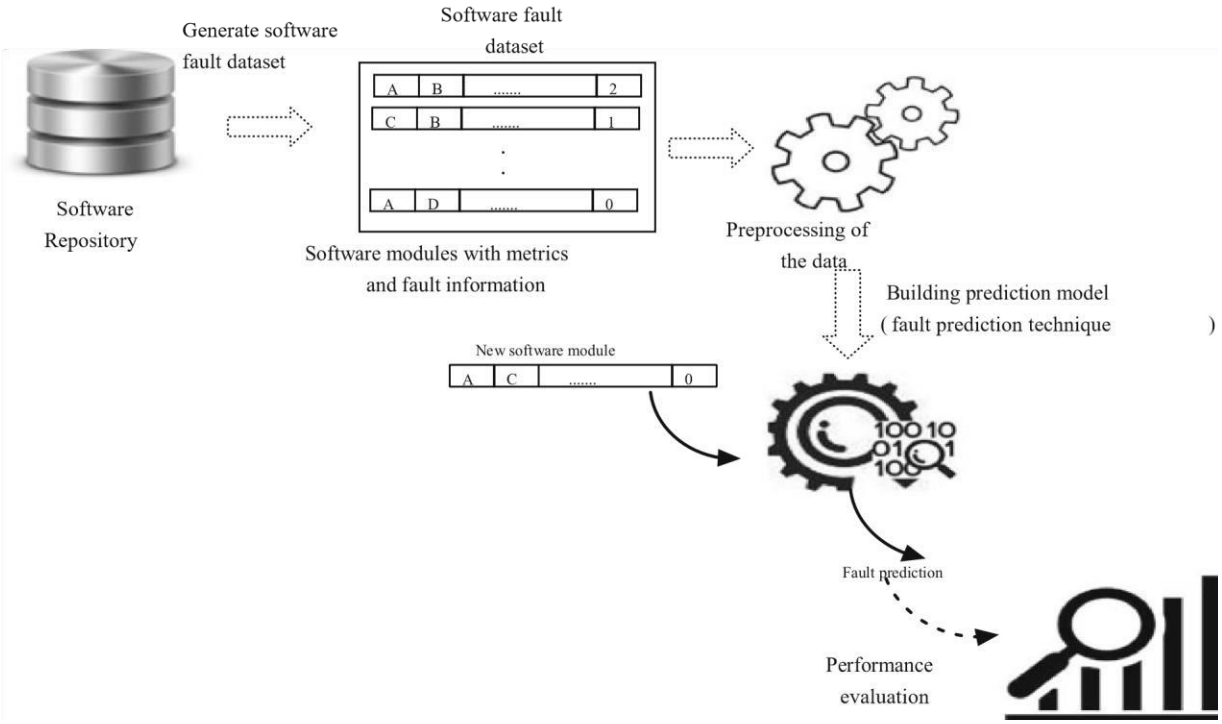
.

2 Data Collection and Preprocessing:

a. Collect a diverse datasetss, including both spam and non-spam.

b. Preprocess the data by cleaning, tokenizing, and transforming text data.

c. Handle missing values and outliers if any.


Generate software fault dataset
Software fault dataset
A B 2
C B 1
A D 0
Software Repository
Software modules with metrics and fault information
Preprocessing of the data
Building prediction model (fault prediction technique)
New software module
A C 0
100 10
01 1
100
Fault prediction
Performance evaluation

3 Literature Survey:

a. Review existing research papers, articles, and documentation related to Software Development and Software Fault prediction.

4 Data Analysis (EDA):

   a. Perform statistical analysis on the dataset.
   b. Visualize the distribution of data.
   c. Explore relevant patterns and correlations.

5 Model Selection:

   a. Choose appropriate machine learning models for classification (e.g., Naïve Bayes, Support Vector Machines, K Nearest).
   b. Split the dataset into training and testing sets.

6 Model Training:

   a. Train the selected models on the training dataset.
   b. Fine-tune hyperparameters for optimal performance.

7 Model Evaluation:

   a. Evaluate models using metrics such as accuracy, precision, recall, and F1-score.
   b. Use a separate validation set to assess generalization.

8 Model Optimization:

   a. Optimize the model based on evaluation results.
   b. Consider techniques like hyperparameter tuning and feature engineering.

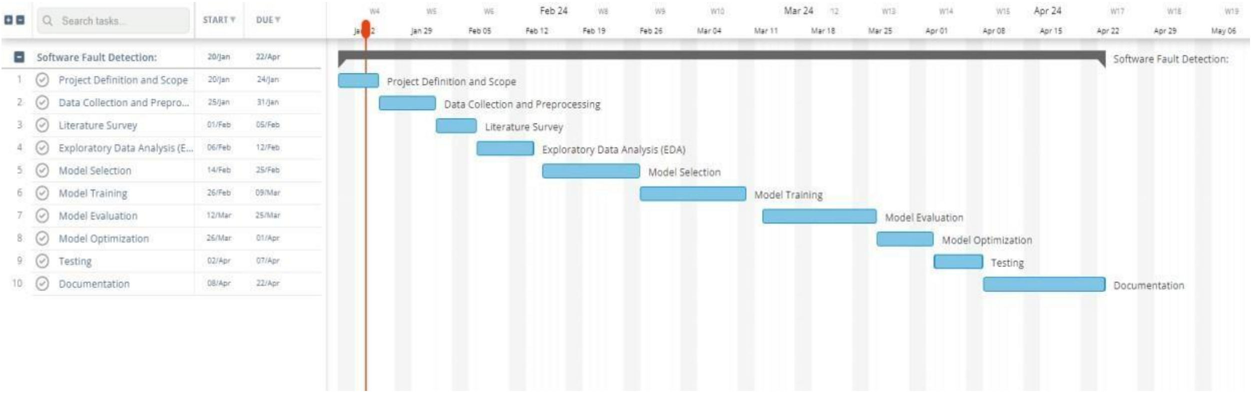
9 Testing:

   a. Conduct thorough testing to identify and fix any issues.
   b. Use diverse test cases to ensure robustness.

10 Documentation:

   a. Document the entire process, including data sources, methodology, and model details.

Gantt Chart



	Search tasks...	START	DUE
	Software Fault Detection:	20/Jan	22/Apr
1	Project Definition and Scope	20/Jan	24/Jan
2	Data Collection and Prepro...	25/Jan	31/Jan
3	Literature Survey	01/Feb	05/Feb
4	Exploratory Data Analysis (E...	06/Feb	12/Feb
5	Model Selection	14/Feb	25/Feb
6	Model Training	26/Feb	09/Mar
7	Model Evaluation	12/Mar	25/Mar
8	Model Optimization	26/Mar	01/Apr
9	Testing	02/Apr	07/Apr
10	Documentation	08/Apr	22/Apr

<u>Facilities required for proposed work</u>

Software Requirements Platform:

1. Operating System: Windows OS
2. Programming Language: PYTHON

Hardware Requirements

1. Processor: INTEL Pentium 4 Processor Core
2. Hard Disk: 40 GB (min)
3. RAM: 256 MB or higher

Performance Requirements

- System can produce results faster on 2GB/4GB of RAM.
- It may take LESS time for peak loads at the main node.
- The system will be available 100% of the time. Once there is a fatal error, the system will provide understandable feedback to the user.

Safety and Security Requirements

The system is designed in modules where errors can be detected and fixed easily. Software Quality Attributes

- **Reliability**: The Client machine will change the status of data indicating successful data transmission.
- **Maintainability:** The system will be developed using the standard software development conventions to help in easy review and redesigning of the system.
- **Portability**: This software is portable to any system with the requirements specified. There must also be a server where the database can be set-up.

References

1. 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.

2. http://www.academia.edu/

3. Software Fault Prediction- A Roadmap By Sandeep Kumar & Santosh Singh Rathore