

EXPLORING ADVANCED TECHNIQUES AND ENHANCING SOFTWARE QUALITY ASSURANCE THROUGH MACHINE LEARNING-BASED FAULT PREDICTION

Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY

in

Software Engineering

by

Sanket Das

(2K22/SWE/17)

Under the Supervision of

Prof. Ruchika Malhotra

Head of Department (Software Engineering)



Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

June, 2024

ACKNOWLEDGEMENTS

I am very thankful to **Prof. Ruchika Malhotra** (Head of Department, Professor, DTU, Department of Software Engineering) and all the faculty members of the Department of Software Engineering at DTU. They all provided us with immense support and guidance for the project. I would also like to express my gratitude to the University for providing us with the laboratories, infrastructure, testing facilities and environment which allowed us to work without any obstructions. I would also like to appreciate the support provided to us by our lab assistants, seniors and our peer group who aided us with all the knowledge they had regarding various topics.

Sanket Das

2K22/SWE/17



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultpur, Main Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

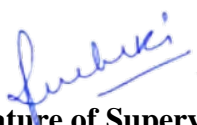
I Sanket Das, Roll no. 2K22/SWE/17 hereby certify that the work being presented in the thesis entitled “**Exploring Advanced Techniques and Enhancing Software Quality Assurance Through Machine Learning-Based Fault Prediction**” in partial fulfilment of the requirements for the award of the Degree of Master of Technology submitted by me to Department of Software Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from 2022 to 2024 under the supervision of Professor Ruchika Malhotra.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute

Place: Delhi


Candidate's Signature

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.


Signature of Supervisor



DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042

CERTIFICATE BY THE SUPERVISOR(s)

Certified that **Sanket Das**(2K22/SWE/17) has carried out their search work presented in this thesis entitled "**Exploring Advanced Techniques and Enhancing Software Quality Assurance Through Machine Learning-Based Fault Prediction**" for the award of **Master of Technology** from Department of Software Engineering, Delhi Technological University, Delhi, under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

Prof. Ruchika Malhotra

SUPERVISOR

HOD and Professor

(Department of Software Engineering)

Place: Delhi

Date:

ABSTRACT

In pursuit of impeccable software quality, crucial for ensuring customer satisfaction and economizing testing efforts, a comprehensive examination of diverse machine learning (ML) techniques was undertaken. Leveraging both established and optimized ML methodologies on an openly accessible dataset, our research aimed at enhancing model performance, particularly in terms of accuracy and precision, surpassing preceding studies. Notably, K-means clustering was employed for class label categorization, followed by the application of classification models on discerned features. Particle Swarm Optimization was instrumental in refining ML models. In our evaluation, we looked at various factors such as precision, recall, F-measure, and different performance error metrics, as well as using a confusion matrix. Our findings showed that both regular machine learning models and enhanced versions performed at their best. Particularly, SVM and its enhanced version achieved high accuracy, with the rates of 99.20% and 99.91%, respectively. The corresponding accuracy rates for NB, RF and the ensemble were it is also impressive with percentages of 94.62, 98.82, and 99%, respectively strong performance. Additionally, the enhanced versions of NB and RF achieved accuracy rates of 94.62% and 99.72%, respectively.

CONTENTS

ACKNOWLEDGEMENTS.....	ii
CANDIDATE’S DECLARATION.....	iii
CERTIFICATE BY THE SUPERVISOR(s).....	iv
ABSTRACT.....	v
CONTENTS.....	vi
LIST OF FIGURES	viii
LIST OF TABLES.....	ix
LIST OF ABBREVIATIONS	x
CHAPTER 1	1
1. INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 Software Fault Prediction	1
1.3 Motivation.....	2
1.4 Objective.....	2
1.5 Thesis Structure	3
CHAPTER 2	4
2. LITERATURE SURVEY	4
2.1. Related works	4
CHAPTER 3	7
3. MODEL DESIGN	7
3.1. Proposed Architecture.....	7
.....	7
3.2. Dataset	8
3.3. Evaluation Metrics.....	10
CHAPTER 4	11
4. METHODOLOGY	11
4.1. Preprocessing of the Dataset.....	11

4.2. Classification	13
4.3. Support Vector Machine (SVM).....	13
4.4. Naïve Bayes (NB).....	14
4.5. Random Forest.....	14
4.6. Ensemble Techniques.....	15
4.7. Feature Selection.....	21
CHAPTER 5	24
5. EXPERIMENTAL RESULT	24
CHAPTER 6	27
6. RESULT DISCUSSION	27
CHAPTER 7	30
7. CONCLUSION AND FUTURE SCOPE.....	30
7.1. Conclusion	30
7.2. Future Scope	30
REFERNCES	31

LIST OF FIGURES

FIGURE NAME	PAGE NUMBER
Fig 1: Proposed Methodology	16
Fig 2: Types of Machine Learning	21
Fig 3: Ensemble Learning Techniques	25
Fig 4: Hard Voting	26
Fig 5: Soft Voting	27
Fig 6: Bagging or Bootstrap Aggregation	28
Fig 7: Boosting	29
Fig 8: Stacking	30
Fig 9: Feature Correlation	31
Fig 10: Evaluation metrics of all models without optimization.	36
Fig 11: Evaluation metrics of all models with optimization	37

LIST OF TABLES

TABLE NAME	PAGE NUMBER
Table 1: Features of the dataset	18
Table 2: Metrics for Assessing Classifiers without Optimization	33
Table 3: Metrics for Assessing Classifiers with Optimization	34

LIST OF ABBREVIATIONS

ABBREVIATIONS	FULL FORM
SFP	Software Fault Prediction
NB	Naïve Bayes
SVM	Support Vector Machine
KNN	K-Nearest Neighbour
DT	Decision Tree
PSO	Particle Swarm Optimization

CHAPTER 1

1. INTRODUCTION

1.1 Introduction

Software testing is part of software development, so it can often require many hours and resources. Delivering software that is totally error-free and satisfies the needs goal of the testing process. To find bugs in software is an expensive way but a necessary process. Since testing is quite an expensive process, it increases the overall project expense. When errors are accurately predicted early on in the process, the software becomes more effective and of higher quality. Accurate defect prediction also helps in maintaining the project within budget.

1.2 Software Fault Prediction

Software Fault Prediction (SFP) is one of the techniques used to bring about an improvement in the quality of software, while at the same time ensuring low testing cost, done through the construction of categorization models based on many machine learning techniques. In the process of software development and maintenance, SFP has played a key role by using ML techniques based on historical data for error prediction, therefore enhancing the process with high-quality developed software within a tight schedule in order to meet customer expectations. SFP is aimed at the delivery of great quality, reliable

software and at the same time toward resource utilization optimization made available throughout the software development life cycle.

1.3 Motivation

As a result, most of the software development organizations would like to estimate and minimize the defects to meet the requirements of the customers and save the testing efforts. At present, SFP is a promising approach that applies machine-learning techniques to develop a classification model—a very effective means for fault estimation. Research work has widely been conducted on the diversity of machine-learning methodologies, such as Decision Trees, Naïve Bayes, multi-layer perceptron, and Random Forests.

1.4 Objective

Recent advancements in machine learning have seen the advent of ensembling strategies and feature selection approaches such as Principal Component Analysis (PCA). In light of the voluminous literature on software metrics for SFP, focusing on the most salient metrics proves pragmatic for accurate defect prediction. SFP leverages Software repository data from the past to assess the dependability and quality of software modules, with software metrics serving as crucial inputs for SFP models. The present study utilizes a publicly available dataset from the Promise Repository, comprising data on diverse applications investigated in NASA from 2005. Following dataset preprocessing and feature selection, K-means clustering facilitates output categorization, followed by the application of machine learning techniques such as Support Vector Machine, Naïve Bayes, and Random Forest, with and without Swarm Intelligence Optimization. An ensemble approach

integrates the results, culminating in a comprehensive analysis and comparison of all models against previous studies. Model performance is evaluated over a range of parameters, such as precision, performance error metrics precision, F-measure, confusion matrix, and recall.

1.5 Thesis Structure

This study's structure is as follows: In Section 2, relevant prior research in the field of software fault prediction is described. Section 3 presents the proposed work and covers the dependent and independent datasets. variables, as well as process and static code measurements. Section 4 presents the research methodology, including the performance evaluation meter utilized, the statistical test used, the classification and ensemble strategies used, and the strategies' implementation. Section 5 displays the results of each model employing each classification strategy and ensemble methodology. Bar graphs and statistical tests are used in Section 6 to display the results discussed.

CHAPTER 2

2. LITERATURE SURVEY

2.1. Related works

The basic models had been worked out in the earlier studies by Jinsheng et al. (2014) [21] and Yong, L.G., Ying, X.L., and Qiong, Z.C. (2014) [27], but they were conducted to support the development of the proposed research based on fundamental methods that used machine learning and CART algorithms. From such seminal works, it is quite evident that computational models can be used, with specific reference to techniques like CART, for the identification of software anomalies. On this history, other researchers then applied many others, such as machine learning techniques and decision trees, in accordance to Naidu and Geethanjali (2013)[12] and Singh and Chug (2017)[36], which reconfirms the constant worth of these techniques in Software Fault Prediction (SFP) applications.

It was at this time, 2015 up to 2018 that witnessed an increase in developing and/or improving innovative approaches to ease the challenges that were associated with the old prediction defect prediction methods. To this Ryad, Arora, Tatarwal, and Sah (2015)[11] accorded room for more research by having a clear review of open issues that surrounded the establishment of innovative approach. Kumudha and Venkatesan (2016) [15] adapted a cost-sensitive radial basis function neural network classifier in order to illustrate the relevance of economic efficiency in model training and implementation. Esteves et al. (2020) [13] covered the gap between theoretical concepts and practical implications so

that thorough insight about machine learning applications in defect prediction could either enlarge the field through actionable insight.

Development offers a wider scope for improving the accuracy of prediction by adopting more sophisticated techniques and methodologies. Ensemble-based aggregate learning methods have been reported in literature earlier for improving the accuracy of prediction. Recently, from work conducted by Faseeha et al. (2021), in the literatures reporting on such kinds of approaches, greater focus has been found. Hybrid approaches, according to Manjula and Florence (2018), combine machine learning approaches with optimization approaches suitable for overcoming the challenges intrinsic within defect prediction. This, in turn, is likely to lead to an inference whereby the field is going towards forms of modeling that are more complex and sophisticated. Further, the deep learning reports of Akimova et al. (2021) and convolutional neural networks over control flow graphs by Phan, Nguyen, and Bui (2017) reach such an inference.

Examples of this are research on bio-inspired algorithms, like the artificial immune network for feature selection by Mumtaz et al. (2021), and further integration with predictive analytics through swarm intelligence by Coelho and Guimaraes (2014). Such comparative analyses as those of Alsaeedi and Khan (2019) [26] and Herbold, Trautsch, and Grabowski (2018) [30] also serve as a benchmark for system performance and at the same time test the effectiveness of cross-project defect prediction approaches.

Other than algorithmic enhancements, making use of cloud computing also presents an effective option for defect prediction systems to acquire better scalability and efficiency; this is confirmed by Ali et al. in 2017. Some other significant contributions in the direction are a feature selection study in ensemble classification frameworks by Iqbal et al. (2019) [10] and a machine learning technique-based detailed analysis of NASA datasets by Iqbal et al. (2019) [32]. In total, software defect prediction reviews how the foundation is the traditional forms of machine learning; after that, the advanced, hybrid, and bio-inspired models are built on that.

New trends in deep learning, ensemble techniques, and cloud computing are so much finer that they have the possibility of shaping up the field further and getting better predictive accuracy. This work, along with the other works, contributes to the development of sophisticated tools ongoing in the area of software defect prediction for enhancement of software quality and reliability. Such integration of methodologies and continuous benchmarking of predictive models from diverse classifiers is a requisite for raising the state of the art in topic areas, moving the research community incrementally closer to a roadmap for future explorations.

This review of the literature is no exception in those ways methodologies have shaped and might point to a number of the future advancements in SFP, a critical area for software quality assurance. This collective effort in this area by researchers shows the importance of machine learning and its derivatives to ensure that the software defect prediction models are precise.

CHAPTER 3

3. MODEL DESIGN

3.1. Proposed Architecture

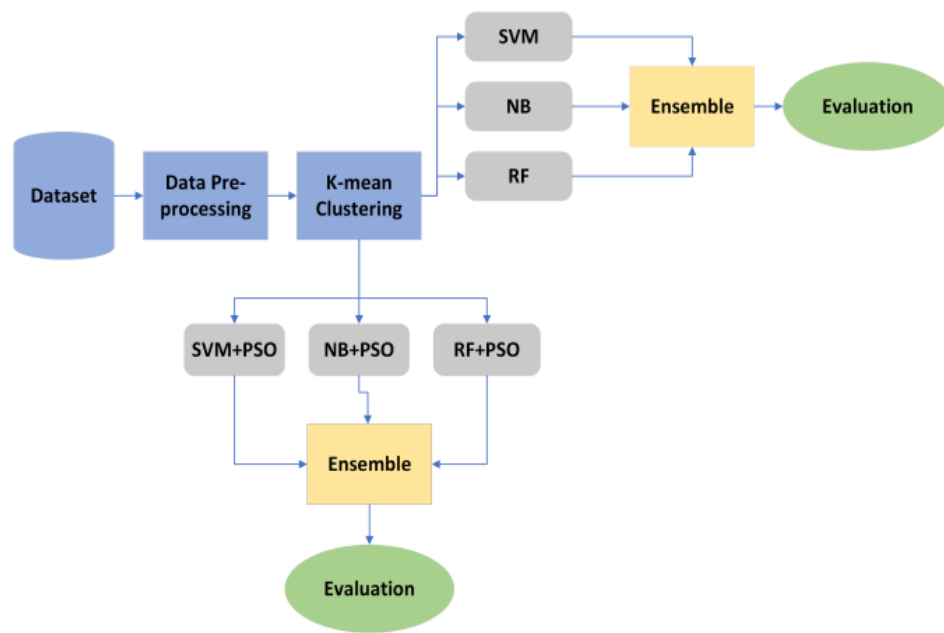


Fig 1: Proposed Methodology

In this regard, integration of this profound development in the field of AI through ML has hence come to be imperative for developing an ML-based model for SFP that helps in supporting software quality and economizing the expenses related to testing. A comprehensive review of literature shows that predicting the defects in software using ML

has a number of challenges because the authors have employed a diverse array of ML algorithms across various datasets that result in discrepancies in performance and accuracy. This paper attempts to develop a prototype framework of SFP analysis primarily working with the cost of tests to reduce and at the same time improve the accuracy of systems. To achieve this, we undertake a rigorous analysis of different ML techniques, coupled with feature selection and clustering methodologies, aimed at optimizing accuracy. Our investigation focuses on attaining superior accuracy utilizing analyzed ML algorithms, particularly on the CM1 dataset, known for its lower accuracy across most techniques. The proposed model architecture, depicted in Figure 1, embodies the culmination of our efforts. Leveraging both established and optimized ML techniques on an open dataset, our approach centers on enhancing dataset accuracy vis-à-vis prior research. Key components include the utilization of K-means clustering for class label categorization, followed by the application of classification models on selected features. Particle Swarm Optimization further fine-tunes ML models to achieve optimal performance.

3.2. Dataset

Our study engaged with the CM1 dataset, sourced from the PROMISE Software Engineering Repository, which is an integral part of the NASA Metrics Data Program (MDP). This particular dataset is associated with a C-language software module designed for NASA's spacecraft instrumentation. It encompasses 498 modules, each exhibiting 22 unique attributes. A meticulous examination of these modules disclosed that 49 exhibited

defects, translating to a defectiveness rate of 9.83%. This finding highlights the criticality of implementing stringent quality control protocols, as a substantial fraction of the modules failed to meet the prescribed quality benchmarks. Conversely, the defect-free status of 449 modules underscores the software's overall structural integrity and operational reliability. These observations are instrumental in advancing the reliability of systems that are crucial to mission success, thereby emphasizing the continual evolution of software engineering methodologies.

TABLE 1: Features of the dataset

Metrics	Description
LOC	Sum of line in the module
iv(g)	Design complexity of each module
ev(g)	Essential complexity of each module
N	Sum of operators and operands existing in the module
V(g)	Cyclomatic complexity of each module
D	Difficulties in each module
B	Effort approximation
L	Program size for each module
V	Volume of each module
I	Intelligence content
E	Error approximation
Locomment	Line of comments in each module
Loblank	Sum of blank lines in each module
uniq_op	Sum of unique operators
uniq_opnd	Sum of unique operand
T	Time determinist
Branchcount	Sum of branch in the software module
total_op	Sum of operators
total_opnd	Sum of operators
Locodeandcomment	Sum of line of code and comments
Defects	Details on whether there is existence of defect or not

The dataset utilized in our investigation is characterized by a diverse array of metrics, which include four McCabe metrics and twelve Halstead measurements, along with additional metrics. The McCabe metrics, which are collected at the method level, provide a straightforward assessment of programming constructs directly from the source code. The Halstead metrics, on the other hand, offer a numerical representation of software complexity and can be readily obtained using various software tools. Additionally, the dataset is enriched with other metrics such as lines of code and comment counts, enhancing our understanding of the software's complexity and maintainability. This multifaceted approach allows for a thorough evaluation of the software's quality and performance.

3.3. Evaluation Metrics

Evaluation metrics encompass accuracy, performance error metrics precision, F-measure, confusion matrix, and recall. analysis. Our rationale for selecting ML techniques is rooted in the variability of findings in existing literature, suggesting room for improvement in accuracy. Thus, our objective is twofold: to enhance performance and accuracy of established ML techniques and meticulously analyze the outcomes for insights and advancements in SFP methodologies.

CHAPTER 4

4. METHODOLOGY

We have used the following steps:

1. Acquire process metrics and static code statistics from publicly available sources.
2. Done some operational processing on the dataset.
3. Clustered the data using K-mean clustering.
4. Choose some classification techniques and ensemble techniques.
5. Used Particle Swarm Optimization on the techniques to compare the results
6. Select performance evaluation metrics to assess the accuracy of the predictions.
7. Analyzed the performance based on the evaluation metrics.

4.1. Preprocessing of the Dataset

After meticulously scrutinising the dataset, it's evident that standardizing the data to a uniform format is essential before applying any machine learning models. With a dataset comprising 498 tuples and 22 features, each column exhibits a wide range of values. For instance, the 'e' column (representing programming effort) varies from a minimum value of 0.0 to a maximum of 1000.0, while the 't' column (representing programming time) ranges from 0.0 to 500.0. Similarly, the 'I' column (representing intelligence) spans from 0.0 to 1.0, indicating considerable diversity among columns.

With such clear differences between the columns, this requires that the data be normalized. One common way of doing this is as follows: this 'typical scaling' transforms the data in such a manner so that it looks similar to a standard normal distribution, at which point fair comparisons of features can subsequently be made. It is a statistical sense to obtain the standardized value of each observation by the formula $\Delta = (K - \mu) / \sigma$, where: Δ is the standardized value K is an observation μ is the mean of sample σ is standard deviation in the sample.

It was also comprehensive when checking through for the missing values on the completeness of the dataset and, hence, assured data integrity. The comprehensive test showed that there was no null in any tuple of the data set, thus it assures completeness.

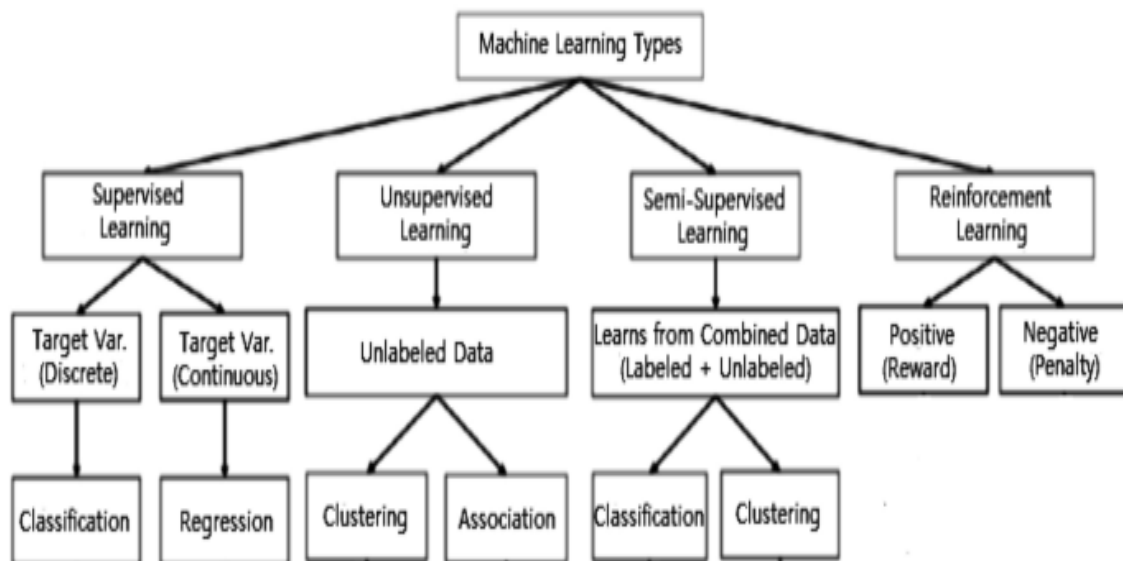


Fig 2: Types of Machine Learning

4.2. Classification

Supervised machine learning methods are then applied on data sets having output class labels and are divided between training and testing sets. In this paper, the training data set comprises 2/3rd of the total and the remaining 1/3rd are assigned to testing. First of all, the training data set which involves output class labels are used to learn the model, a step which is then followed by testing the unseen data which does not contain the class labels. The Machine Learning algorithm within our analysis constitutes mainly three separate components for classification: a Linear Support Vector Machine (SVM), Gaussian Naive Bayes (NB), and Random Forest (RF). A Stacked Generalization model is also applied to combine the results of Naive Bayes as the base model with Support Vector Machine and Random Forest as member models.

4.3. Support Vector Machine (SVM)

The Support Vector Machine model is for two-class classification tasks and treats data with special care, showing a lot of efficiency, especially with smaller datasets. The Support Vector Machine maximizes the margins separating classes by drawing a line between the data points, conflicting with the rationale for the increase in accuracy of the classification rule. Our implementation uses linear Support Vector Classifier (SVC) with fixed random state of 42.

4.4. Naïve Bayes (NB)

Naïve Bayes predicts it all: Naïve in the way it proceeds by making the assumption that the occurrence of a certain feature associated with a class is in no way related to the presence or absence of other features. Since this makes calculations easier, this in turn helps the algorithm to predict easily. Some of the most common variations of NB include:

1. **Gaussian NB:** Assumes a Gaussian (Normal) Distribution for Numeric Features.
2. **Multinomial NB:** This is to be used when classification will have to assume that it's performed with discrete feature counts, even word counts for document classification.
3. **Bernoulli Naive Bayes:** This model is similar to Multinomial, but it assumes binary features; it is often applied in binary text classification problems.

In contrast, the Naive Bayes algorithm bases on the Bayes theory and classifies data by making use of the different probabilities and likelihoods within the datasets. It is particularly effective in those phenomena with weak interrelation among the dataset attributes. In this work, the Gaussian Naive Bayes classifier is put to work with a fixed random state number 42 and standardization.

4.5. Random Forest

Random Forest, on the other hand, harnesses ensemble learning principles, amalgamating multiple decision trees to refine predictions. By employing 1000 decision trees, our Random Forest model optimizes performance, with a fixed random state of 42 to ensure reproducibility and consistency.

4.6. Ensemble Techniques

The objective is to aggregate the prediction results of various learning approaches so that the overall performance of the decision is enhanced. The ensemble model improves the performance of the individual model for example it improves the performance of the decision tree by reducing variance in the model. They are classified as either homogeneous or heterogeneous ensembles. In a homogeneous ensemble, similar type of learning techniques like bagging, boosting, and others are employed. Different types of learning techniques are used in heterogeneous ensembles. We built a defect prediction model using voting, stacking, bagging, and boosting in this study. Ensemble techniques are machine learning methods that combine the predictions of multiple individual models, known as base models or weak learners, to improve the overall predictive performance. By leveraging the diversity and collective wisdom of multiple models, ensemble techniques aim to achieve better generalization, reduce overfitting, and enhance prediction accuracy. Fig. 4.2 depicts the categories of ensemble techniques. Among all the applied methodologies, the ensemble techniques have been proved to apply across several classes of problems. Applications of these methods cut across in machine-learning tasks including classification, regression, and anomaly detection. These methods have also demonstrated competitive success in both competitions and real-world cases. The selection of a specific ensemble technique depends on the problem at hand, data characteristics, and the preferred trade-offs between performance and interpretability.

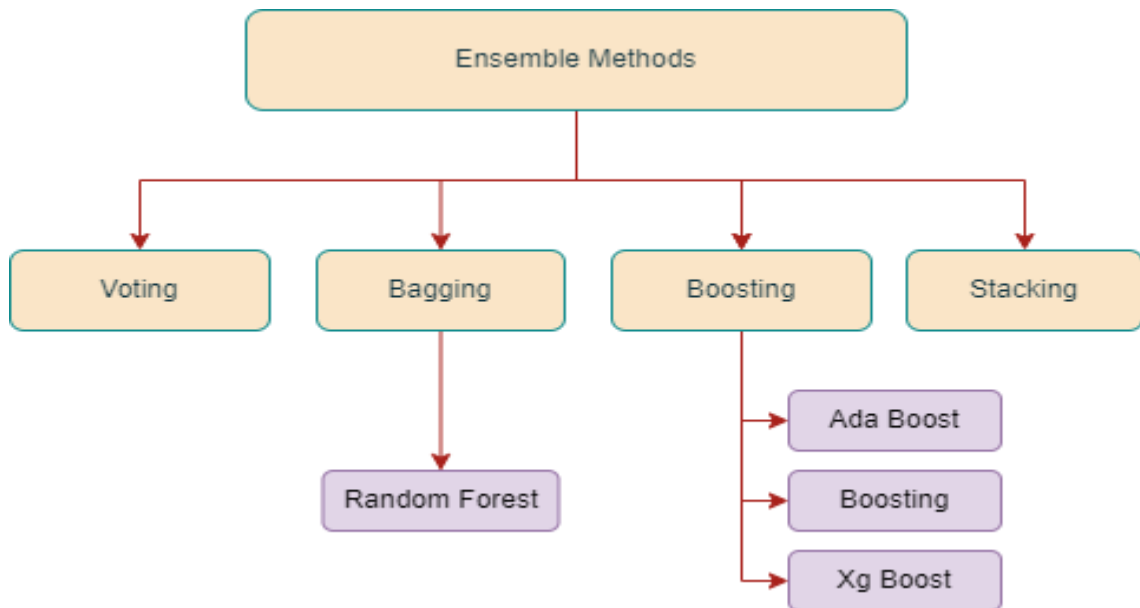


Fig 3: Ensemble Learning Techniques

4.6.1. Voting

Voting might be the act of using either multiple models to make the predictions or multiple classifiers, and the final decision is made by voting on individual decisions. Voting can also be an extra form of ensemble learning that has a goal of increasing the general robustness and accuracy of the predictions made. There are two types of voting:

1. Hard Voting

Hard voting, also known as deterministic voting, considers only the expected class labels of models. On the other hand, the final prediction is made based on the most

occurring label among the predictions. Fig. shows the diagram of the hard voting system.

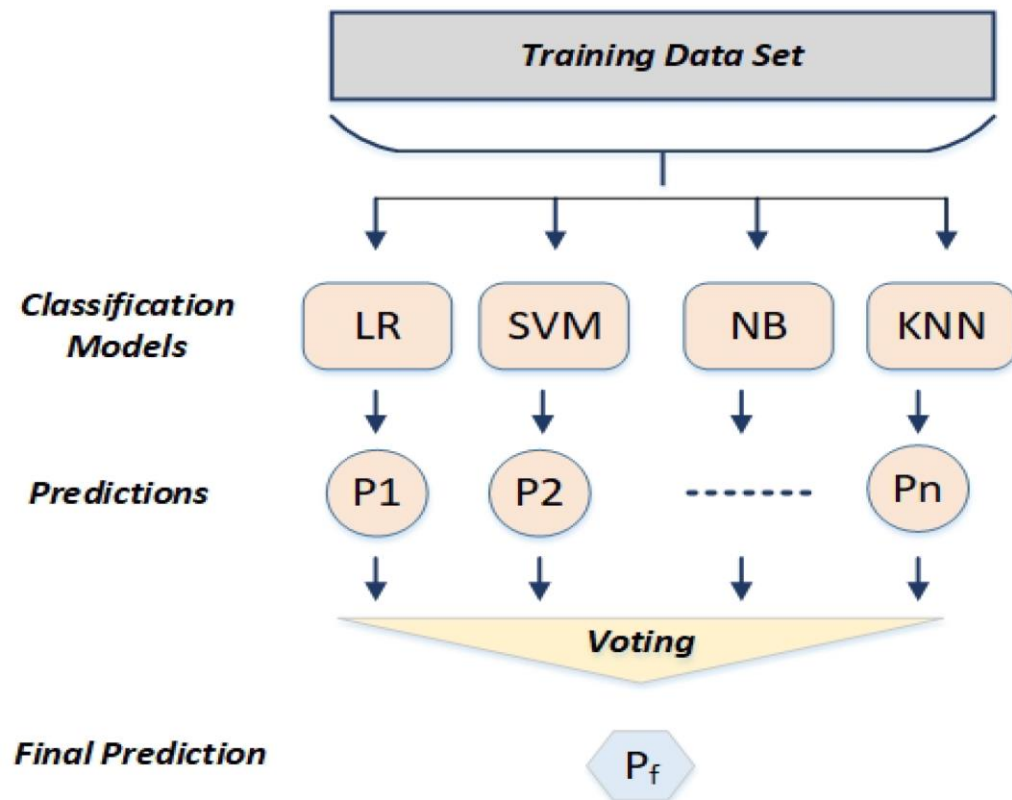


Fig 4: Hard Voting

2. Soft Voting

Probabilistic voting or soft voting refers to the class probability, or the confidence scores, attributed to the class label by each model. This process, then, aggregates

the individual models' class probabilities and selects the class label for which the average probability is maximum as the final prediction.

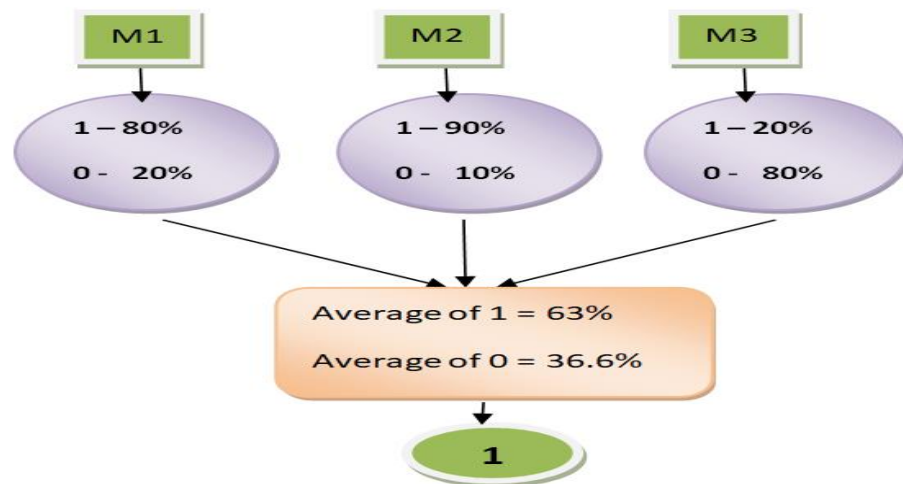


Fig 5: Soft Voting

4.6.2. Bagging

Bagging is a short abbreviation for bootstrap aggregating. It is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. To accomplish a good generalization ability, bagging trains multiple models based on various training data subsets. It is an effective ensemble learning technique used to boost prediction accuracy and stability.

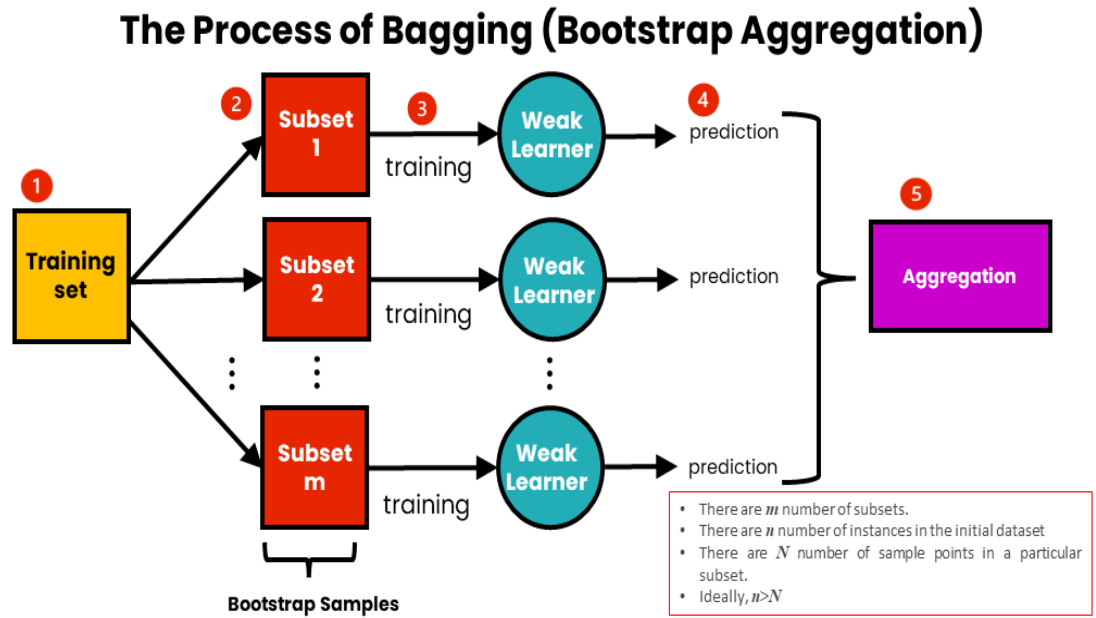


Fig 6: Bagging or Bootstrap Aggregation

4.6.3. Boosting

Boosting is a machine learning technique utilized to construct a strong predictive model from the combination of a few weak models; these are often termed base learners or weak learners. In contrast to bagging, in which the base models are independently trained, boosting trains the base models in a sequential adaptive manner, as it trains a new base model and puts emphasis on its gradient mainly for those cases where the previous base models were misclassified. Boosting is a way to increase predicted accuracy by giving higher weights to points that are not well classified. XGBoost, Gradient Boosting, Adaboost, and LightGBM are some of the popular boosting algorithms.

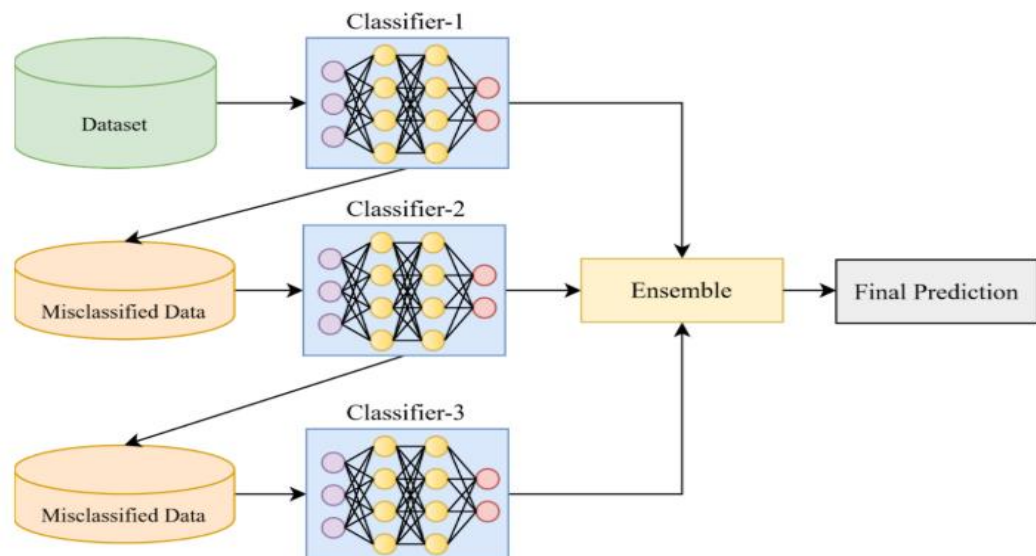


Fig 7: Boosting

4.6.4. Stacking

In machine learning, the term "stacking" describes a method where several models also referred to as base models or learners are combined to enhance prediction outcomes. It is an example of ensemble learning, which uses the advantages of various models to produce predictions that are more accurate. The base models in a stacking ensemble are trained on the same dataset, and a meta-learner, also known as a stacking model, is used to combine the predictions of the base models. The meta-classifier learns how to effectively integrate the basic models' predictions to generate the final prediction.

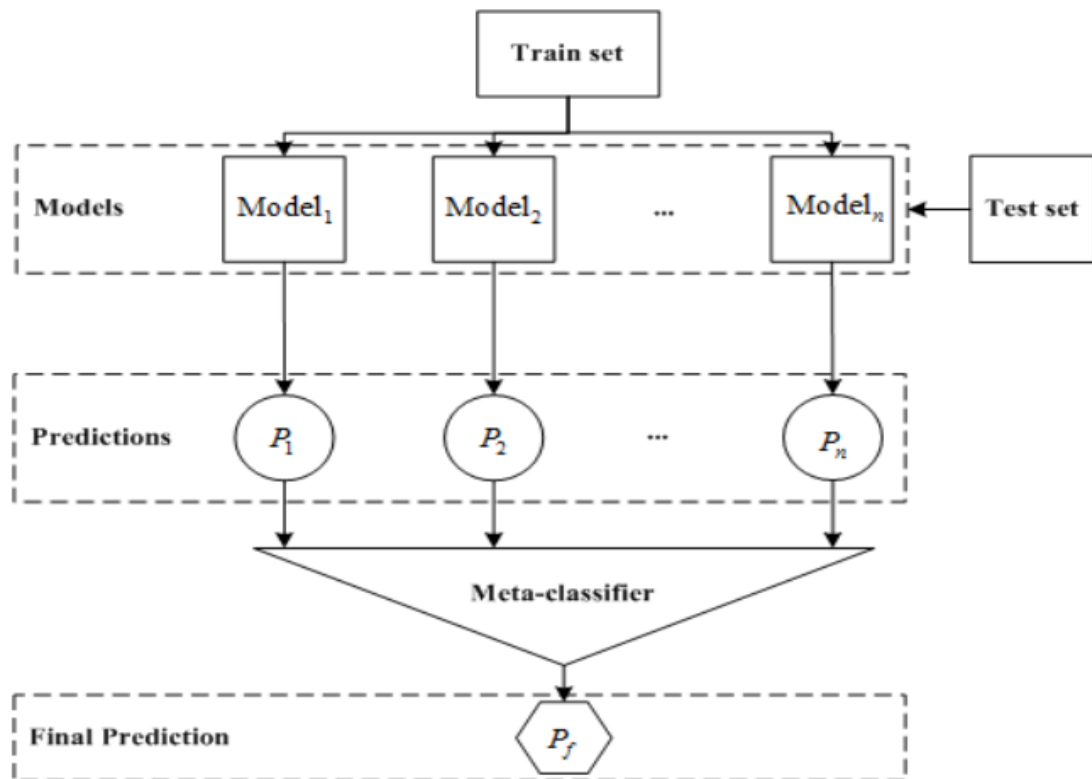


Fig 8: Stacking

4.7. Feature Selection

The process feature selection is vital to the improvement of prediction models. by curtailing the number of features utilized during training and testing phases. In this study, we employ the variance inflation factor approach and correlation method to gauge the significance of values and assess multicollinearity among features post dataset preparation. Features exhibit positive correlation when A rise in one feature is accompanied by a surge in others, or vice versa, while no correlation is observed when

changes in one feature have no bearing on others. Conversely, negative correlation occurs when an increase in one parameter corresponds with a decrease in another, and vice versa.

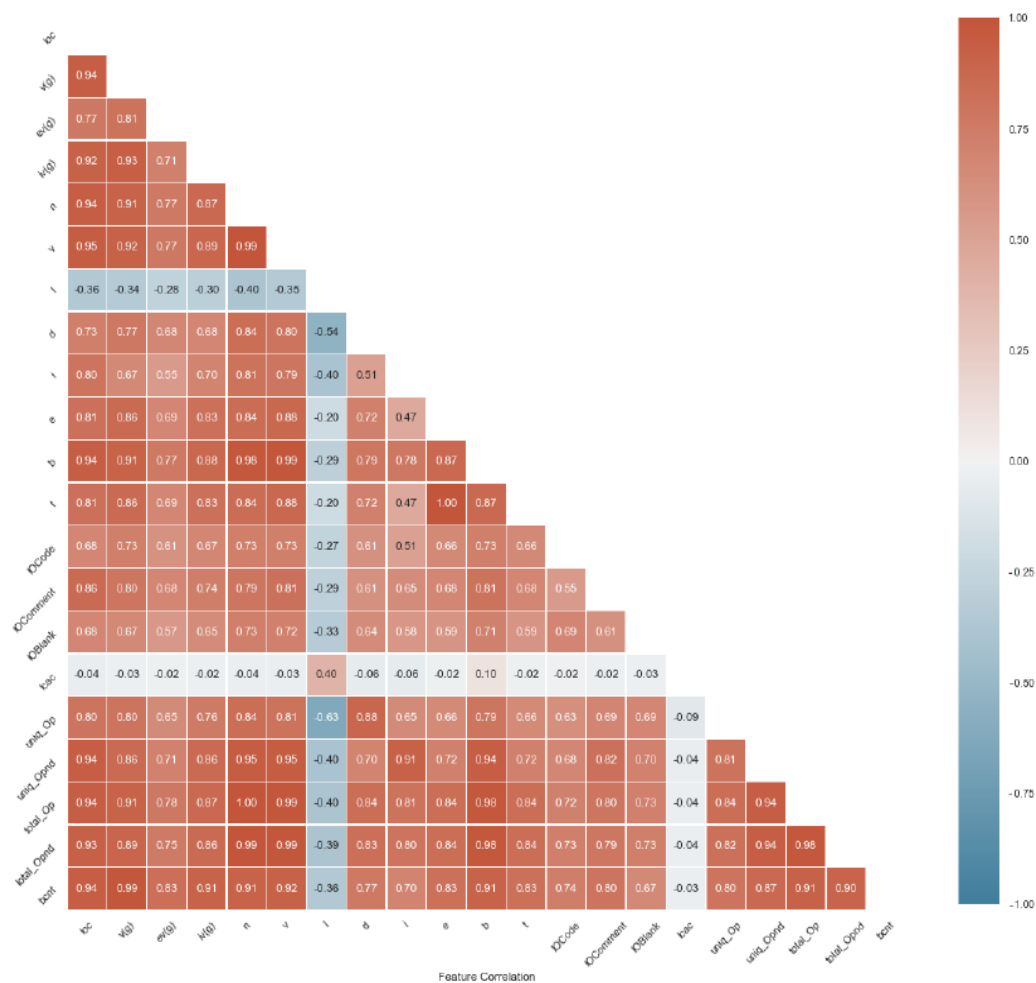


Fig 9: Feature Correlation

Visualizing the correlation between all features in the dataset, as depicted in Figure 2, enables us to discern the top 10 attributes with either negative or negligible associations, devoid of class labels. This meticulous selection is underpinned by the imperative of reducing computational overhead and circumventing overfitting challenges, thereby

bolstering model performance. In our research, we leverage both unsupervised machine learning techniques, and supervised machine learning, particularly classification, for predictive analysis. This hybrid approach ensures comprehensive coverage and effectiveness in addressing the complexities inherent in predictive modeling.

CHAPTER 5

5. EXPERIMENTAL RESULT

In this section, the result obtained experimentally using ensemble learning and classification algorithms on each of the datasets is presented. The graphical and tabular form represents the result based on the CM1 dataset experimentally obtained using different ML techniques. There are striking graphical representations developed containing comparisons between strategies with insights into the performances of each one. Comparisons are also made between optimized and non-optimized methods in both strategies.

TABLE 2: Metrics for Assessing Classifiers without Optimization

Dataset	Evaluation Measures	SVM	NB	RF	Ensemble
	Accuracy	99.2	94.62	98.82	99
CM1	Precision	100	100	100	100
	Recall	90.9	50	82.2	82.3
	F-measure	95.2	66.7	90.9	90.9

TABLE 3: Metrics for Assessing Classifiers with Optimization

Dataset	Evaluation Measures	SVM	NB	RF	Ensemble
	Accuracy	99.91	94.73	99.87	99.21
	Precision	99.7	100	100	99
CM1	Recall	100	92.9	99.5	97.5
	F-measure	96	67.3	91.1	84.8

Tables 2 and 3 represent all-inclusive evaluation metrics for every ML methods investigated after optimization and without optimization, respectively. In the performance of all models, there is praiseworthy performance; however, Support Vector Machine and its optimized version are far better than all the rest models in both settings. Specifically, SVM yields a near-perfect precision value in predicting class 1 observations; as a result of misclassification by some class 0 observations, the recall value is slightly less than it can be.

On the other hand, NB gives poor performance, but at least it gives an accuracy rate for higher-dimensional data. The same results are also shown by RF and ensemble techniques because the technique of RF is using based on the ensemble technique by combining the decision trees. The model of RF gives a precision of 100% and an accuracy of 98.7%, a

nice design to make the predictions. The class 1 instances are also predicted in class 0 while an error in prediction of class 0 is almost negligible.

Briefly, all the models showed their proficiency, among which SVM and optimized SVM gave continuously better results across all the metrics of evaluation. On the other hand, it is evident that NB lacks in low-dimensional data sets while RF and ensemble techniques show good results, particularly in terms of precision and accuracy.

CHAPTER 6

6. RESULT DISCUSSION

All results for these non-optimized models of machine learning are depicted in Figure 10, including the ensemble approach. The best classifying metric is prominently found in precision across all algorithms, indicated where each of the models correctly predicts class 1 instances. However, due to the suitability of handling large quantities of data or high-dimensional datasets, the performance of the NB model greatly drops in f-measure and recall. Both the results of the Random Forest (RF) model and the ensemble are competitive with each other, but the Support Vector Machine (SVM) model outperformed by a high margin in all the metrics employed to evaluate model performance.



Fig 10: Evaluation metrics of all models without optimization.

Figure 11 presents the aggregate results of all predictive models utilizing Particle Swarm Optimization (PSO). Despite achieving commendable evaluation metrics, including precision, all models, including the ensemble technique, exhibit a slight dip in performance compared to their unoptimized counterparts. This observation underscores the notion that optimization solutions are particularly advantageous when confronted with larger datasets. Notably, improvements in f-measure and recall are discernible with optimization, suggesting enhanced model performance in scenarios with increased dataset sizes.

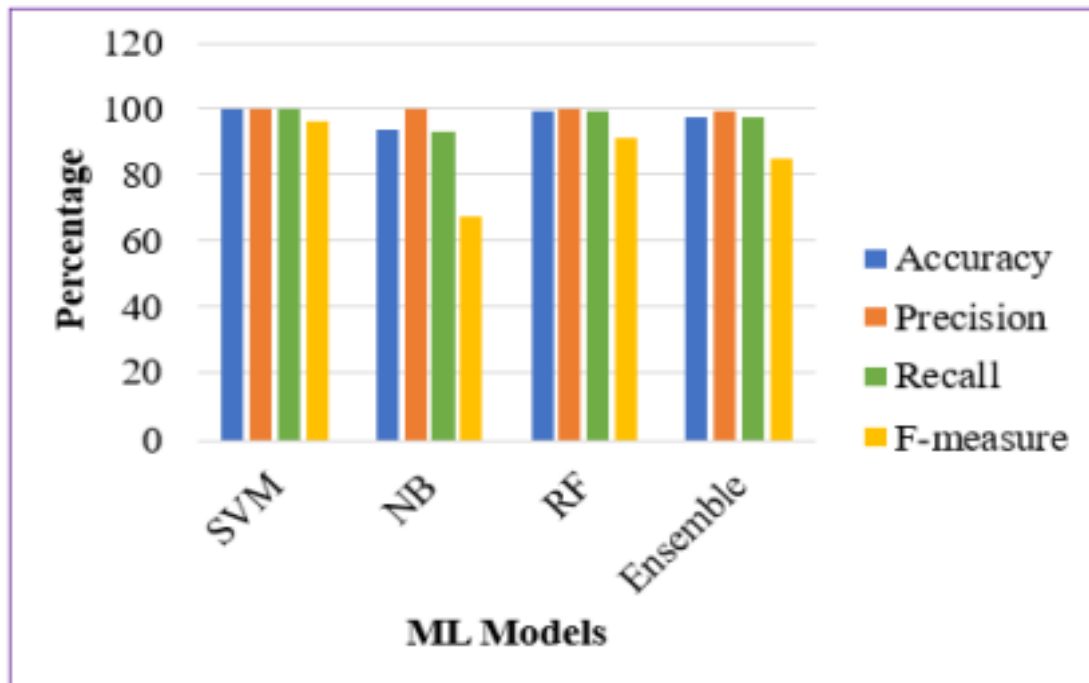


Fig 11: Evaluation metrics of all models with optimization.

In summary, while all models, both with and without optimization, achieve satisfactory evaluation metrics, the unoptimized SVM model consistently outshines its counterparts. However, optimization solutions, while beneficial for larger datasets, result in a marginal reduction in performance across all models, albeit with noticeable improvements in recall and f-measure metrics.

CHAPTER 7

7. CONCLUSION AND FUTURE SCOPE

7.1. Conclusion

A wide range of models have been developed and are being investigated in the field of software fault prediction. Many of those identify faulty software codes or programs using static code metrics. In this project the efficiency of process metrics or variable have been analyzed using classification and ensemble methods merging with Particle Swarm Optimization (PSO) on the basis of recall, precision, accuracy, f-measures, bar graphs.

It can be concluded that the combined outcomes of any predictive model that makes use of Particle Swarm Optimization (PSO). Even though all models—including the ensemble technique achieves satisfactory evaluation measures like precision, their performance is marginally lower than that of their unoptimized counterparts.

7.2. Future Scope

Ensemble and classification methods can also be used to examine other process parameters. Regression analysis can be used to forecast the number of defects while taking into account the same datasets utilized in this study. Rather than bugs, effort or maintainability can alternatively be thought of as dependent variables.

REFERENCES

- [1] Singh, P.D.; Chug, A. Software defect prediction analysis using machine learning algorithms. In Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science and Engineering-Confluence IEEE, Noida, India, 12–13 January 2017; pp. 775–781.
- [2] Li, R.; Zhou, L.; Zhang, S.; Liu, H.; Huang, X.; Sun, Z. Software Defect Prediction Based on Ensemble Learning. In Proceedings of the 2019 2nd International Conference on Data Science and Information Technology, Seoul, Republic of Korea, 19–21 July 2019; pp. 1–6.
- [3] Maddipati, S.; Srinivas, M. Machine learning approach for classification from imbalanced software defect data using PCA and CSANFIS. *Mater. Today Proc.* 2021, 52, 471.
- [4] Perreault, L.; Berardinelli, S.; Izurieta, C.; Sheppard, J. Using classifiers for software defect detection. In Proceedings of the 26th International Conference on Software Engineering and Data Engineering, Sydney, Australia, 2–4 October 2017; pp. 2–4.
- [5] Li, Z.; Jing, X.Y.; Zhu, X. Progress on approaches to software defect prediction. *IET Softw.* 2018, 12, 161–175.
- [6] Hammouri, A.; Hammad, M.; Alnabhan, M.; Alsarayrah, F. Software bug prediction using machine learning approach. *Int. J. Adv. Comput. Sci. Appl.* 2018, 9, 78–83.

- [7] Akimova, E.N.; Bersenev, A.Y.; Deikov, A.A.; Kobylkin, K.S.; Konygin, A.V.; Mezentsev, I.P.; Misilov, V.E. A Survey on Software Defect Prediction Using Deep Learning. *Mathematics* 2021, 9, 1180.
- [8] Coelho, R.A.; Guimaraes, F.D. Applying Swarm Ensemble Clustering Technique for Fault Prediction Using Software Metrics. In *Proceedings of the 2014 13th International Conference on Machine Learning and Applications, Date of Conference, Detroit, MI, USA, 3–6 December 2014*; pp. 356–361.
- [9] Arar, Ö.F.; Ayan, K. Software defect prediction using cost-sensitive neural network. *Appl. Soft Comput.* 2015, 33, 263–277.
- [10] Iqbal, A.; Aftab, S.; Ullah, I.; Bashir, M.S.; Saeed, M.A. A feature selection based ensemble classification framework for software defect prediction. *Int. J. Mod. Educ. Comput. Sci.* 2019, 11, 54.
- [11] Arora, I.; Tetarwal, V.; Saha, A. Open issues in software defect prediction. *Procedia Comput. Sci.* 2015, 46, 906–912.
- [12] Naidu, M.S.; Geethanjali, N. Classification of defects in software using decision tree algorithm. *Int. J. Eng. Sci. Technol.* 2013, 5, 1332.
- [13] Esteves, G.; Figueiredo, E.; Veloso, A.; Viggiano, M.; Ziviani, N. Understanding machine learning software defect predictions. *Autom. Softw. Eng.* 2020, 27, 369–392.
- [14] Mumtaz, B.; Kanwal, S.; Alamri, S.; Khan, F. Feature selection using artificial immune network: An approach for software defect prediction. *Intell. Autom. Soft Comput.* 2021, 29, 669–684.

- [15] Kumudha, P.; Venkatesan, R. Cost-sensitive radial basis function neural network classifier for software defect prediction. *Sci. World J.* 2016, 2016, 2401496.
- [16] Ali, M.M.; Huda, S.; Abawajy, J.; Alyahya, S.; Al-Dossari, H.; Yearwood, J. A parallel framework for software defect detection and metric selection on cloud computing. *Clust. Comput.* 2017, 20, 2267–2281.
- [17] Hassan, F.; Farhan, S.; Fahiem, M.A.; Tauseef, H. A Review on Machine Learning Techniques for Software Defect Prediction. *Tech. J.* 2018, 23, 63–71.
- [18] Cetiner, M.; Sahingoz, O.K. A Comparative Analysis for Machine Learning based Software Defect Prediction Systems. In *Proceedings of the 2020 11th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 1–3 July 2020; pp. 1–7
- [19] Rawat, M.; Dubey, S.K. Software Defect Prediction Models for Quality Improvement: A Literature Study. *Int. J. Comput. Sci.* 2012, 9, 288–296.
- [20] Fenton, N.E.; Neil, M. A critique of software defect prediction models. *IEEE Trans. Softw. Eng.* 1999, 25, 675.
- [21] Jinsheng, R.; Ke, Q.; Ying, M.; Guangchun, L. On Software Defect Prediction Using Machine Learning. *J. Appl. Math.* 2014, 2014, 785435.
- [22] Tua, F.M.; Sunindyo, W.D. Software Defect Prediction Using Software Metrics with Naïve Bayes and Rule Mining Association Methods. In *Proceedings of the 2019 5th International Conference on Science and Technology (ICST)*, Yogyakarta, Indonesia, 30–31 July 2019; pp. 1–5.

- [23] Manjula, C.; Florence, L. A Deep neural network based hybrid approach for software defect prediction using software metrics.
- [24] Faseeha, M.; Taher, M.; Nasser, T.; Shabib, A.; Munir, A.; Muhammad, A.; Sagheer, A.; Tariq, R. Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review. *IEEE Access* 2021, 9, 98754–98771.
- [25] Paramshetti, P.; Phalk, D.A. Software defect prediction for quality improvement using hybrid approach. *Int. J. Appl. Innov. Eng. Manag.* 2015, 4, 99–104.
- [26] Alsaeedi, A.; Khan, M.Z. Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study *J. Softw. Eng. Appl.* 2019, 12, 85–100.
- [27] Yong, L.G.; Ying, X.L.; Qiong, Z.C. Research of software defect prediction based on CART. *Int. J. Adv. Comput. Sci. Appl.* 2014, 602, 3871–3876.
- [28] Manjula, C.; Florence, L. Hybrid approach for software defect prediction using machine learning with optimization technique. *Int. J. Comput. Inf. Eng.* 2018, 12, 28–32.
- [29] Phan, A.V.; Nguyen, M.L.; Bui, L.T. Convolutional neural networks over control flow graphs for software defect prediction. In *Proceedings of the 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, Boston, MA, USA, 6–8 November 2017; pp. 45–52.
- [30] Herbold, S.; Trautsch, A.; Grabowski, J. A comparative study to benchmark cross-project defect prediction approaches. *IEEE Trans. Softw. Eng.* 2018, 44, 811–833.
- [31] Aljamaan, H.; Alazba, A. Software defect prediction using tree-based ensembles. In *Proceedings of the 16th ACM International Conference on Predictive Models and*

- Data Analytics in Software Engineering, Virtual USA, 8–9 November 2020; pp. 1–10.
- [32] Iqbal, A.; Aftab, S.; Ali, U.; Nawaz, Z.; Sana, L.; Ahmad, M.; Husen, A. Performance analysis of machine learning techniques on software defect prediction using NASA datasets. *Int. J. Adv. Comput. Sci. Appl.* 2019, 10, 300–308.
- [33] Ali, M.M.; Huda, S.; Abawajy, J.; Alyahya, S.; Al-Dossari, H.; Yearwood, J. A parallel framework for software defect detection and metric selection on cloud computing. *Clust. Comput.* 2017, 20, 2267–2281.
- [34] Tua, F.M.; Sunindyo, W.D. Software Defect Prediction Using Software Metrics with Naïve Bayes and Rule Mining Association Methods. In *Proceedings of the 2019 5th International Conference on Science and Technology (ICST)*, Yogyakarta, Indonesia, 30–31 July 2019; pp. 1–5.
- [35] Li, Z.; Jing, X.Y.; Zhu, X. Progress on approaches to software defect prediction. *IET Softw.* 2018, 12, 161–175.
- [36] Singh, P.D.; Chug, A. Software defect prediction analysis using machine learning algorithms. In *Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science and Engineering-Confluence IEEE*, Noida, India, 12–13 January 2017; pp. 775–

LIST OF ACCEPTED PAPERS

[1] Malhotra, R. Das, S. (2024) Enhancing Software Quality Assurance through Machine Learning-Based Fault Prediction. In: 1st International Conference on Advances in Computing, Communication and Networking- ICAC2N (IEEE Xplore, Scopus Indexed) (Accepted).

5/21/24, 10:42 AM Gmail - Acceptance Notification 1st IEEE ICAC2N-2024 & Registration: Paper ID 518 @ ITS Engineering College, Greater Noida



Sanket Das <dassanket11@gmail.com>

Acceptance Notification 1st IEEE ICAC2N-2024 & Registration: Paper ID 518 @ ITS Engineering College, Greater Noida

1 message

Microsoft CMT <email@msr-cmt.org>

13 May 2024 at 00:15

Reply-To: "Dr. Vishnu Sharma" <vishnu.sharma@its.edu.in>
To: Sanket Das <dassanket11@gmail.com>

Dear Sanket Das,
Delhi Technological University
Greetings from ICAC2N-2024 ...!!!!
Congratulations.....!!!!!!

On behalf of the ICAC2N-2024 organising Committee, we are delighted to inform you that the submission of "Paper ID- 518 " titled " Enhancing Software Quality Assurance through Machine Learning-Based Fault Prediction " has been accepted for presentation and further publication with IEEE at the ICAC2N- 24. All accepted papers will be submitted for inclusion into IEEE Xplore subject to meeting IEEE Xplore's scope and quality requirements.

Registration/Fee Payment related details are available at <https://icac2n.in/register>.

For early registration benefit please pay your fee and complete your registration by clicking on the following Link: <https://forms.gle/E7RuvuQQPXPZQnJU6> by 20 May 2024.

Note:

1. All figures and equations in the paper must be clear.
2. Final camera ready copy must be strictly in IEEE format available on conference website.
3. Transfer of E-copyright to IEEE and Presenting paper in conference is compulsory for publication of paper in IEEE.
4. If plagiarism is found at any stage in your accepted paper, the registration will be cancelled and paper will be rejected and the authors will be responsible for any consequences. Plagiarism must be less than 15% (checked through Turnitin).
5. Change in paper title, name of authors or affiliation of authors will not be allowed after registration of papers.
6. Violation of any of the above point may lead to rejection of your paper at any stage of publication.
7. Registration fee once paid will be non refundable.

If you have any query regarding registration process or face any problem in making online payment, write us at icac2n.ieee@gmail.com.

Regards,
Organizing committee
ICAC2N - 2024

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our [Privacy Statement](#).

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052



Need to the Indexing of the conference

1 message

Dr. Vishnu Kumar Sharma <dean.cse@its.edu.in>
To: Sanket Das <dassanket11@gmail.com>

Tue, 28 May, 2024 at 9:38 pm

Dear Sanket,

This is to inform you that the IEEE International Conference on Advances in Computing, Communication and Networking (ICAC2N) will be held on December 16-17, 2024, at ITS Engineering College. The conference has a Record Number #63387 and is IEEE Xplore compliant with ISBN No. 979-8-3503-5681-6 and it is a Scopus-indexed conference, providing a prestigious platform for researchers and academics to share their work.

Regards

Dr Vishnu Sharma
Convener, IEEE International Conference -ICAC2N-2024
Professor and Dean CSE
ITS Engineering College, Greater Noida
M-7835878146

On Tue, 28 May, 2024, 20:51 Sanket Das, <dassanket11@gmail.com> wrote:

Sir, it will be very helpful if let me know the Indexing of the conference ICAC2N - 2024.
Regards,

ITS Engineering College
46, Knowledge park-III, Greater Noida- 201310
Ph: 0120- 2331000, 2331001
Connect with us:-



- [2] Malhotra, R. Das, S. (2024) Exploring Advanced Techniques for Software Defect Prediction: A Comprehensive Review. In: 1st International Conference on Advances in Computing, Communication and Networking- ICAC2N (IEEE Xplore, Scopus Indexed) (Accepted).

5/21/24, 10:41 AM

Gmail - Acceptance Notification 1st IEEE ICAC2N-2024 & Registration: Paper ID 115 @ ITS Engineering College, Greater Noida



Sanket Das <dassanket11@gmail.com>

Acceptance Notification 1st IEEE ICAC2N-2024 & Registration: Paper ID 115 @ ITS Engineering College, Greater Noida

1 message

Microsoft CMT <email@mrs-cmt.org>

13 May 2024 at 00:15

Reply-To: "Dr. Vishnu Sharma" <vishnu.sharma@its.edu.in>
To: Sanket Das <dassanket11@gmail.com>

Dear Sanket Das,
Delhi Technological University
Greetings from ICAC2N-2024 ...!!!!
Congratulations.....!!!!!!

On behalf of the ICAC2N-2024 organising Committee, we are delighted to inform you that the submission of "Paper ID- 115 " titled " Exploring Advanced Techniques for Software Defect Prediction: A Comprehensive Review " has been accepted for presentation and further publication with IEEE at the ICAC2N- 24. All accepted papers will be submitted for inclusion into IEEE Xplore subject to meeting IEEE Xplore's scope and quality requirements.

Registration/Fee Payment related details are available at <https://icac2n.in/register>.

For early registration benefit please pay your fee and complete your registration by clicking on the following Link: <https://forms.gle/E7RuvuQQPxPZQnJU6> by 28 May 2024.

Note:

1. All figures and equations in the paper must be clear.
2. Final camera ready copy must be strictly in IEEE format available on conference website.
3. Transfer of E-copyright to IEEE and Presenting paper in conference is compulsory for publication of paper in IEEE.
4. If plagiarism is found at any stage in your accepted paper, the registration will be cancelled and paper will be rejected and the authors will be responsible for any consequences. Plagiarism must be less than 15% (checked through Turnitin).
5. Change in paper title, name of authors or affiliation of authors will not be allowed after registration of papers.
6. Violation of any of the above point may lead to rejection of your paper at any stage of publication.
7. Registration fee once paid will be non refundable.

If you have any query regarding registration process or face any problem in making online payment, write us at icac2n.ieee@gmail.com.

Regards:
Organizing committee
ICAC2N - 2024

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our [Privacy Statement](#).

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052



Need to the Indexing of the conference

1 message

Dr. Vishnu Kumar Sharma <dean.cse@its.edu.in>
To: Sanket Das <dassanket11@gmail.com>

Tue, 28 May, 2024 at 9:38pm

Dear Sanket,

This is to inform you that the IEEE International Conference on Advances in Computing, Communication and Networking (ICAC2N) will be held on December 16-17, 2024, at ITS Engineering College. The conference has a Record Number #63387 and is IEEE Xplore compliant with ISBN No. 979-8-3503-5681-6 and it is a Scopus-indexed conference, providing a prestigious platform for researchers and academics to share their work.

Regards
Dr. Vishnu Sharma
Convener, IEEE International Conference -ICAC2N-2024
Professor and Dean CSE
ITS Engineering College, Greater Noida
M-7835878146

On Tue, 28 May, 2024, 20:51 Sanket Das, <dassanket11@gmail.com> wrote:
Sir, It will be very helpful if let me know the Indexing of the conference ICAC2N - 2024.
Regards,

ITS Engineering College
46, Knowledge park-III, Greater Noida- 201310
Ph: 0120- 2331000, 2331001
Connect with us:-

