SOFTWARE DEFECT PREDICTION USING SELECTED MACHINE LEARNING TECHNIQUES

SAKA KAYODE KAMIL, ARO TAYE, HIKMAT BELLO-ABDULMUMIN, ABDULRAUF TOSHO AND GBOLAGADE MORUFAT DAMOLA, TAOFIK ABIODUN AHMED
Department of Computer Science, Al-Hikmah University
Email: Kamilsaka67@gmail.com, taiwo774@gmail.com, hikmatbello@gmail.com
Abdtosh@gmail.com, dammyconsult@gmail.com, at.ahmed@kwaracails.edu.ng

Abstract
A common challenge in machine learning is selecting the optimal model hyperparameters. However, almost all studies in Software Defection Prediction (SDP) using machine learning models did not perform hyperparameter tuning to obtain the optimal model hyperparameters. As a result, this study examined selected Machine Learning (ML) techniques on a freely available data set. The purpose of the research was to improve the model performance in terms of accuracy, precision and f1 score of the dataset compared to previous research. As previous investigations show, the accuracy can be further improved. For this purpose, this study employed Particle Swarm Optimization for the feature selection. Further, this study applied classification models such as Support Vector Machine (SVM), Artificial Neural Network (ANN), Random Forest (RF) and Recursive Partitioning Trees (RPT) to classified features. This study evaluated the performance of models through precision, accuracy, recall, f-measure, performance error metrics, and a confusion matrix. The results indicate that all the selected ML models achieve the maximum results; however, the random forest classifier and Recursive Partitioning Trees models outperformed with the highest achieved accuracy, 98% and 99.80%, respectively. The accuracy of SVM and ANN approaches are 83.55% and 82.60%, respectively. In this way, this study achieved maximum accuracy compared to previous studies.

Keywords: Software Defect Prediction, Hyoerparameters, Particle Swarm Optimization, Recursive Partitioning Trees, Machine Learning.

Introduction

Software Defect Prediction (SDP) is an integral aspect of the Software Development Life-Cycle (SDLC) and is a crucial task in software engineering that can be utilized to maintain software quality. Identifying software defects at an early stage can result in decreased development expenses, rework efforts, and more reliable software. (Akbar et al., 2024). Ensuring the reliability and quality of these systems is paramount, yet the complexity of software development often leads to defects. Software defects can range from minor bugs to critical errors that can cause system failures, leading to financial losses, compromised security, and reduced user satisfaction (Ceylan, Kutlubay & Bener, 2016). Consequently, in today's competitive software landscape, companies can't afford to ship buggy applications. By embracing SDP, they can build higher-quality software more efficiently, while delighting customers and staying ahead of the competition (Hammad, Alqaddoumi & Al-Obaidy, 2019).

As a result, software engineers must now focus on improving their ability to detect and prevent software defects. Software Defect Prediction (SDP) is a crucial technique that identifies potential software defects before they occur. In software engineering, SDP is an important and challenging task. Better software quality and reduced development costs are both linked to early defect detection in software development (Prabha & Shivakumar, 2020).

In recent times, there has been a widespread utilization of machine learning models to identify flaws in software systems. This trend can be attributed to the capability of machine learning

models to autonomously discern patterns within datasets, thereby facilitating the detection of software defects. SDP serves to enhance software quality by employing diverse machine learning methods to construct classification models, enabling efficient testing procedures. And also software systems are integral to the functioning of businesses, industries, and daily life which influences various machine learning techniques to predict the likelihood of defects in software modules (Ali *et al.,* 2024).

The quality of results obtained by feature selection techniques is very dependent on datasets. The significance of this study lies in its potential to enhance the reliability and robustness of software systems by predicting earlier defects and significantly lowering the costs associated with fixing bugs later in the software development lifecycle (Anjali, *et al.,* 2023). It is also important to improve the allocation of testing resources by focusing on the most defect-prone areas and providing valuable insights into the most effective machine-learning techniques for software defect prediction. This study Reduced Debugging and Testing effort by predicting where defects are likely to occur, software development teams can focus their testing efforts on critical areas, saving time and resources that would otherwise be spent on exhaustive testing. This study enhances software quality, reduces costs, mitigates risks, and supports innovation, and ultimately leads to more efficient and reliable software development processes.

Literature Review

Angga and Elish (2024) examined hybrid approach for effective software defect prediction: Integrating Hybrid Grey Wolf and Particle Swarm Optimization for Enhanced Feature Selection with Popular Gradient Boosting Algorithm. The study utilizes 13 NASA MDP datasets. These datasets are divided into testing and training data using 10-fold cross-validation. After data is divided, the SMOTE technique is employed in training data. The results showed that the performance was at its best, with an average accuracy of 92%. In order to improve performance in a cross-software defect prediction evaluation scenario, synthetic data can also be investigated.

Aimen *et al.,* (2023) examined a machine learning-based software defect prediction analysis. K-means clustering was utilized in this study to categorize the class labels. Additionally, classification models where used on specific features. Particle Swarm Optimization is utilized to optimize ML models. Precision, accuracy, recall, f-measure, performance error measures, and a confusion matrix were used to assess the models' performance. The findings show that while the ML and optimized ML models produce the best outcomes, the SVM and improved SVM models performed best, with accuracy levels of 99% and 99.80%, respectively. The accuracy of NB, Optimized NB, RF, Optimized RF and ensemble approaches are 93.90%, 93.80%, 98.70%, 99.50%, 98.80% and 97.60, respectively. The dataset to which these models are applied is small. The plan is to expand the dataset in the future and attempt to examine various ensemble classifier types after using data balancing approaches. This is because these techniques allow us to enhance error measures and achieve the best possible results.

Karedla and Satyanada (2023) investigated the use of machine learning algorithms for software defect prediction. Seven well-known machine learning methods for defect prediction were the subject of our experimental investigation, which used the PROMISE datasets at both the method and class levels. Naïvebayes and Random Forest have the best performance for class-level datasets, with 85% and 88%, respectively, according to the results obtained. Data imbalance and reduced dimensionality are the study's two main problems. The oversampling and feature-selection methods aim to resolve these two issues. In subsequent research, we

will examine classification methods to address the imbalance problem of datasets for defect prediction.

Madadi, Aravind and Kamal (2023) investigated the use of machine learning algorithms for software defect prediction. In this investigation, machine learning techniques such as artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P are employed. The algorithms were tested on a publicly available dataset, and the error rates of the predicted values were evaluated using a number of statistical techniques. The ML algorithm's accuracy, precision, recall, F-measure, and AUC are 74.24 percent, 72%, 79%, and 0.81, respectively. The results suggest that machine learning algorithms are useful tools for predicting future software errors. The random tree algorithm obtained the lowest F-measures error rate for both (79% and 85%) training sets while employing percentage split testing mode.

Mengtian, Songlin, Yue, and Xu, (2022) examined Software Defect Prediction Model Based on Complex Network and Graph Neural Network. The proposed model is tested on the PROMISE dataset, using two graph convolution methods, based on the spectral domain and spatial domain in the graph neural network. The investigation indicated that both convolution methods showed an improvement in various metrics, such as accuracy 86.6%, F-measure 85.8% and MCC (Matthews's correlation coefficient) 87.5%. respectively. So it needs high improvement in term of speeding up the computational time and accuracy.

Alaa *et. al.,* (2019) Dynamic Detection of Software Defects Using Supervised Learning Techniques. They experiment with different parameter values for the classifiers and explore the usefulness of employing dimensionality reduction techniques, such as Principle Component Analysis (PCA), and Ensemble Learning techniques. Additionally, using PCA did not have a noticeable impact on prediction systems performance while parameter tuning positively impact classifies' accuracy, especially with Artificial Neural Network (ANN). The best results are obtained by using Ensemble Learning methods such as Bagging (95.1% accuracy with the Mozilla dataset) and Voting (93.79% accuracy with the kc1 dataset).

Table 1: Review of Related works

Paper	Authors/ Year	Tittle	Methodology	Result and Limitation
1	Angga et al., (2024)	examined hybrid approach for effective software defect prediction	Hybrid Grey Wolf and Particle Swarm Optimization for Enhanced Feature Selection. The study utilizes 13 NASA MDP datasets	The results showed that the performance was at its best, with an average accuracy of 92%. The study suffer for high feature dimension
2	Aimen et al., (2023)	propose software defect prediction analysis using machine learning techniques.	K-means clustering for the categorization of class labels	The SVM and optimized SVM models outperformed with the highest

			and support vector machine for classification	achieved accuracy, 96% and 95.80%, respectively These models are applied to a limited dataset. In the future, we will increase the size of the dataset and try to analyze different types of ensemble classifiers after applying data balancing techniques because due to balancing technique we can improve error measure as well to get maximum results
3	Karedla and Satyanada (2023).	Examined Software Defect Prediction using Machine Learning Algorithms	Our experimental study was conducted on seven popular machine learning techniques for predicting defects by using the PROMISE datasets at both method-level and class-level.	The obtained results conclude that Naïvebayes has the highest performance for class-level datasets with 85% and Random Forest performance has 88% for class-level datasets. The two major issues in this study are the data imbalance and reduced dimensionality. The

				oversampling and feature-selection methods aim to resolve these two issues.
4	Madadi, Aravind and Kamal (2023)	examined Software Defects Prediction using Machine Learning Algorithms	ML algorithms used in this inquiry include ANNs (artificial neural networks), RF (random forest), RT (random tree), DT (decision table), LR (linear regression), GP (gaussian processes), SMOreg, and M5P. A publicly accessible dataset was utilised to test the algorithms	The ML Algorithm through standard deviation has an accuracy of 74.24 percent, a precision of 72%, a recall of 79%, an F-measure of 75%, and an AUC of 0.81. In contrast, the RT algorithm obtained the lowest F-measures error rate for both (79% and 85%) training sets while employing percentage split testing mode.
5	Mengtian, Songlin, Yue, & Xu, (2022)	examined Software Defect Prediction Model Based on Complex Network and Graph Neural Network.	The proposed model is tested on the PROMISE dataset, using two graph convolution methods, based on the spectral domain and spatial domain in the graph neural network.	The investigation indicated that both convolution methods showed an improvement in various metrics, such as accuracy 86.6%, F-measure 85.8% and MCC (Matthews's correlation coefficient)

	87.5%. respectively. So it needs high improvement in term of speeding up the computational time and accuracy.

Methodology

The software defect detection model contains several steps. Figure 1 presents the steps followed in the development of the model. The first step is dataset acquisition from the NASA database. The particle swarm optimization was employed as a feature selection algorithm. Lastly, the classification of the selected features was achieved by passing them to selected learning algorithms like SVM, RT,
RF and ANN in predicting software defects.
NASA database

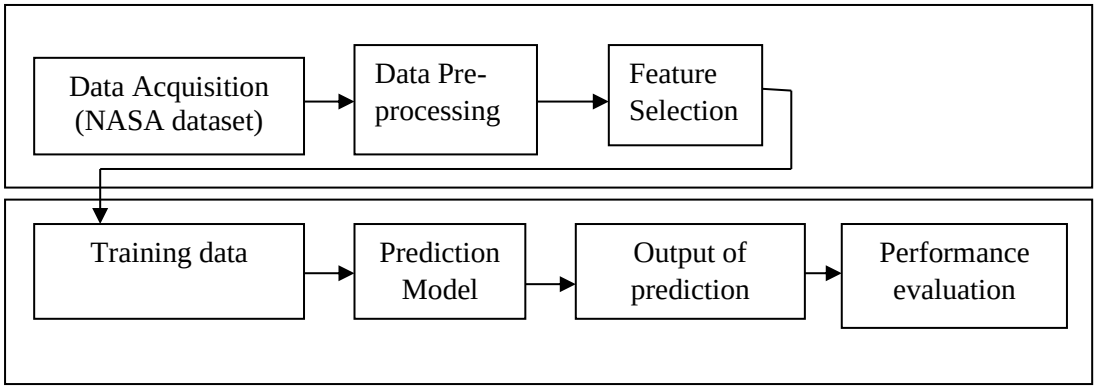
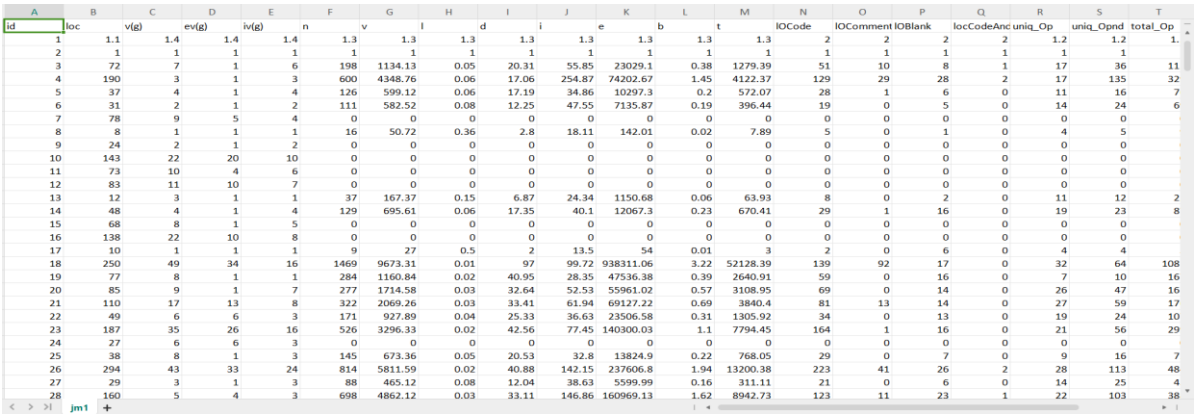


Figure 1. Research Framework

Dataset Acquisition

This dataset, titled "NASA Dataset" was taken from the Kaggle website. There are 23 total columns in the dataset, one of which is the class column that needs to be predicted. The total dataset is made up of 10,885 rows of data points. The target feature is the column 'defects' which can be predicted as 'Defects' or 'Non-Defects', 'True' or 'False' while the other 22 columns are the input features. A screenshot of the *NASA* dataset is shown in Figure 2.



id	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	b	t	lOCode	lOComment	lOBlank	locCodeAnc	uniq_Op	uniq_Opnd	total_Op
1	1.1	1.4	1.4	1.4	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	2	2	2	2	1.2	1.2	1.
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
3	72	7	1	6	198	1134.13	0.05	20.31	55.85	23029.1	0.38	1279.39	51	10	8	1	17	36	11
4	190	3	1	3	600	4348.76	0.06	17.06	254.87	74202.67	1.45	4122.37	129	29	28	2	17	135	32
5	37	4	1	4	126	599.12	0.06	17.19	34.86	10297.3	0.2	572.07	28	1	6	0	11	16	7
6	31	2	1	2	111	582.52	0.08	12.25	47.55	7135.87	0.19	396.44	19	0	5	0	14	24	6
7	78	9	5	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	8	1	1	1	16	50.72	0.36	2.8	18.11	142.01	0.02	7.89	5	0	1	0	4	5	
9	24	2	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	143	22	20	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11	73	10	4	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12	83	11	10	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
13	12	3	1	1	37	167.37	0.15	6.87	24.34	1150.68	0.06	63.93	8	0	2	0	11	12	2
14	48	4	1	4	129	695.61	0.06	17.35	40.1	12067.3	0.23	670.41	29	1	16	0	19	23	8
15	68	8	1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	138	22	10	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
17	10	1	1	1	9	27	0.5	2	13.5	54	0.01	3	2	0	6	0	4	4	
18	250	49	34	16	1469	9673.31	0.01	97	99.72	938311.06	3.22	52128.39	139	92	17	0	32	64	108
19	77	8	1	1	284	1160.84	0.02	40.95	28.35	47536.38	0.39	2640.91	59	0	16	0	7	10	16
20	85	9	1	7	277	1714.58	0.03	32.64	52.53	55961.02	0.57	3108.95	69	0	14	0	26	47	16
21	110	17	13	8	322	2069.26	0.03	33.41	61.94	69127.22	0.69	3840.4	81	13	14	0	27	59	17
22	49	6	6	3	171	927.89	0.04	25.33	36.63	23506.58	0.31	1305.92	34	0	13	0	19	24	10
23	187	35	26	16	526	3296.33	0.02	42.56	77.45	140300.03	1.1	7794.45	164	1	16	0	21	56	29
24	27	6	6	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
25	38	8	1	3	145	673.36	0.05	20.53	32.8	13824.9	0.22	768.05	29	0	7	0	9	16	7
26	294	43	33	24	814	5811.59	0.02	40.88	142.15	237606.8	1.94	13200.38	223	41	26	2	28	113	48
27	29	3	1	3	88	465.12	0.08	12.04	38.63	5599.99	0.16	311.11	21	0	6	0	14	25	4
28	160	5	4	3	698	4862.12	0.03	33.11	146.86	160969.13	1.62	8942.73	123	11	23	1	22	103	38

Figure 2: NASA Dataset

The NASA dataset properties are based on Halstead Metrics, numerical data and they may be easily gathered with any piece of software (Promise Software Engineering Repository, 2021)

Data Preprocessing

From dataset analysis, it is concluded that the dataset needs to be transformed to a standard format before applying any ML models, as there are 498 tuples and 23 features in the dataset. Thus, for this purpose, a standard scaling technique is used in this study to standardize the data set. It arranges data in a standard normal distribution. Mathematically, the standard scalar Z can be determined as:

$$\mathrm{m} = \frac{(n-u)}{v} \tag{3.1}$$

where m is an observation, u is the training samples' mean, and the training samples' standard deviation is v. In the dataset, we checked for null values, but there were no null values in any tuple.

SMOTE Techniques for Overcoming Imbalance Data Issues

Smote is an oversampling technique in which false samples are manufactured for the minority class. This strategy helps to overcome the over-fitting problem caused by random oversampling. It concentrates on the feature space to develop new instances by using interpolation between positive instances that are close together. The initial value of N is the overall value of oversampling data. It is usually chosen in such a way that the binary distribution of classes is 1:1. But depending on the situation, that might be dialled back. The loop then starts by selecting a positive category instance at random. The KNNs for that instance are obtained next. Finally, N is chosen to interpolate new synthetic instances from among these K instances. The distance between the feature vector and its neighbours is determined to achieve this using any distance metric. Now, this difference is multiplied by any random value in [0,1] and is added to the previous feature vector. Figure 3 display Oversampling Imbalanced Data

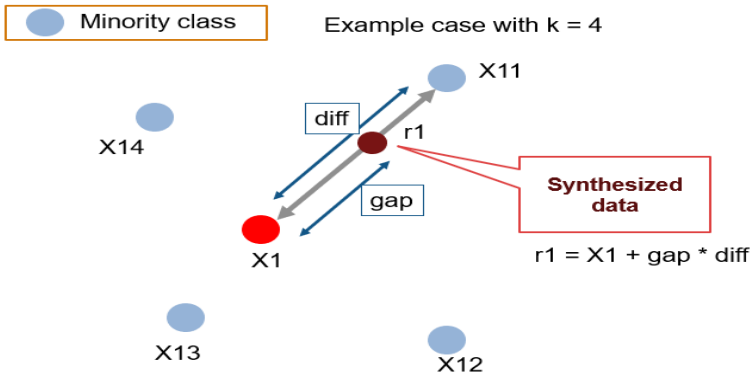


Figure 3: Oversampling Imbalanced Data (Source: Github, 2020).

Feature Selection

The feature selection method is used to reduce the number of features utilized in a predictive model's training and testing. The particle swarm optimization approach is utilized in this research work to identify the significance of the values of a feature after preparing the dataset. Particle Swarm Optimization (PSO) is a popular metaheuristic algorithm that can be effectively used for this purpose. The dataset was subjected to Particle Swarm Optimization (PSO) to choose features. Here's an overview of how PSO can be applied to feature selection in the context of software defect prediction. The algorithm is shown in Figure 4

PSO Algorithm

Step1: Start
Step 2:. Initialization
Initialize Parameter
Initialize Population
a) Initialize Position (X_i) randomly for each particle.
b) Initialize Velocity (V_i) randomly for each particle.
Step 3: Evaluate Fitness f(X^t_i)
a) Calculate the Fitness Value for Each Particle
b) If Fitness Value is better than Best Fitness value (g Best).
c) Then
d) Set New value as new (g Best)
e) Choose the Particle with the Best Fitness Value as guest
Step 4:b For each particle calculate the Velocity and Position
a) Calculate particle position by: $X^{t+1}_i = X^t_i + V^t_i * t$
b) Calculate Velocity by**:** $v^i_{k+1} = \omega v^i_k + c_1 r_1(\text{xBest}^t_i - x^t_i) + c_2 r_2(\text{gBest}^t_i - x^t_i)$
Step 5: Evaluate Fitness f(X^t_i)
Find Current Best [gBest]
Step 6: Updated = t + 1
Step 7: Output gBest& X^t_1
Step 8: End

Figure 4: PSO Algorithm

3.5 Classification

Supervised Machine Learning (ML) is applied to data with output class labels. There are two portions to the data set: training and testing. The training dataset makes up 67% of the total, whereas the testing dataset makes up just 33%. First, the data are provided with output class

labels for training purposes, and then unseen data with no output class labels are provided for testing. We used four ML classification models for analysis, i.e., SVM, RF, ANN and RT.

Support Vector Machine

SVM is a supervised ML model mostly applied to data with two classes as output (Grishma & Anjali, 2018). SVM models perform better with high speed if the dataset has limited data. The SVM model is employed in this work to predict the output class label. Figure 5 displays the SVM Algorithm as

Algorithm1: Support Vector Machine Algorithm with Particle Swarm Optimization

1. First step:
2. employ PSO to select important features
3. compute the importance score for the feature
4. $F = {\sigma_1^2}/{\sigma_2^2}$
5. select the threshold that maximizes the model's performance
6. Reduce the variables
7. Then:
8. Make use of the SVM linear-kernel function$\emptyset(X\mathrm{i})$
9. establish the separation hyperplane
10. $\mathrm{W}^{\mathrm{T}}\ \emptyset(X\mathrm{i}) + b = 0$
11. Sort the data into non-fraud and fraudulent classes.

Figure 5: SVM Algorithm (Source: Naoufal & Nourdeen, 2020).

Random Forest

The RF is a classification model that employs the notion of ensemble learning, which entails combining numerous classifiers to improve the outcome. The RF model comprises several DTs that are applied to subsets of the data set and then averaged to determine performance measures. The number of trees used has a significant impact on accuracy and other measures. The model improvement, however, becomes constant after a certain number of trees. Knowing the right amount of trees is crucial for training purposes. In this case, 1000 trees are employed, and the random state is 42. As a classifier, Random Forest is utilized.

Artificial Neural Network

This employs multiple layers of neurons to capture complex patterns in data. Configure layers, neurons per layer, activation functions, and optimization algorithms. Reduced Error Pruning (REP) Tree is a decision tree with pruning to prevent overfitting.

Recursive Partitioning Trees

Recursive Partitioning Trees are a type of decision tree used in statistical modeling. and machine learning. They work by dividing the data into subsets based on feature values to improve prediction accuracy. This method is useful for classification and regression tasks. It can handle both categorical and numerical data

Performance Evaluation Metrics

There are several performance evaluation metrics for evaluating the effectiveness of the model. These evaluation metrics include:

1.1.1

1.1.2 Accuracy

It measures the proportion of total correct predictions (both true positives and true negatives) out of all predictions made. It can be calculated from the Equation (3.2)

$$\text{Accuracy} = \frac{(\text{TP}+\text{TN})}{(\text{TP}+\text{TN}+\text{FP}+\text{FN})} \quad (3.2)$$

TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

1.1.3 Precision

It assesses the accuracy of positive predictions and can be calculated from Equation (3.3)

$$\text{Precision} = \frac{\text{TP}}{(\text{TP}+\text{FP})} \quad (3.3)$$

1.1.4 Recall (Sensitivity)

It indicates the ability of the model to detect all actual positives, measuring the percentage of true positives identified correctly. It can be calculated from the equation.

$$\text{Recall} = \frac{\text{TP}}{(\text{TP}+\text{FN})} \quad (3.4)$$

1.1.5 F1-Score

It is the harmonic mean of precision and recall. It provides a balance between the two in cases where equilibrium is needed for effective performance evaluation. It can be calculated from the equation.

Result and Discussion

Experimental Analysis

The Machine Learning (ML) classification experiments in this document were performed on a Windows laptop with an Intel(R) Core (TM) i5-2410 M processor, 6 GB of primary storage and. The ML model was implemented using the Python programming language. Python is commonly used in predictive analytics and data science projects involving both qualitative and quantitative data. The results of the experiments on the NASA dataset using various ML techniques are reported.

4.1. Performance of SVM Model with PSO

Table 2: Performance Evaluation of SVM Model with PSO

Models	ACCURACY(%)	PRECISION (%)	RECALL (%)	F1-Score(%)
SVM (Training set)	82.60	83.86	12.63	21.95
SVM (Testing set)	81.07	56.14	7.64	13.45

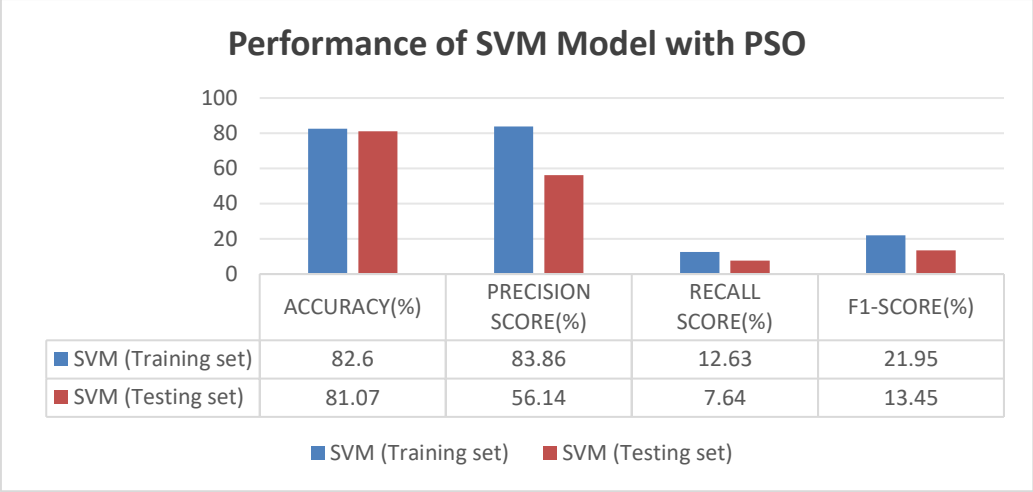


Figure 6: Performance of SVM with PSO Feature Selection

Figure 6 illustrates the performance of the SVM model with training and testing data. The accuracy, precision score, recall and F1-score of the SVM model for training data are 82.6%, 83.86%, 12.63% and 21.95% respectively.

Table 3: Performance evaluation of the RPT Model with PSO

Models	ACCURACY (%)	PRECISION (%)	RECALL S (%)	F1-SCORE (%)
RPT (Training set)	98.93	99.75	94.72	91.17
RPT (Testing set)	75.33	36.47	37.95	37.19

4.3 Performance of Recursive Partitioning Trees Model with PSO

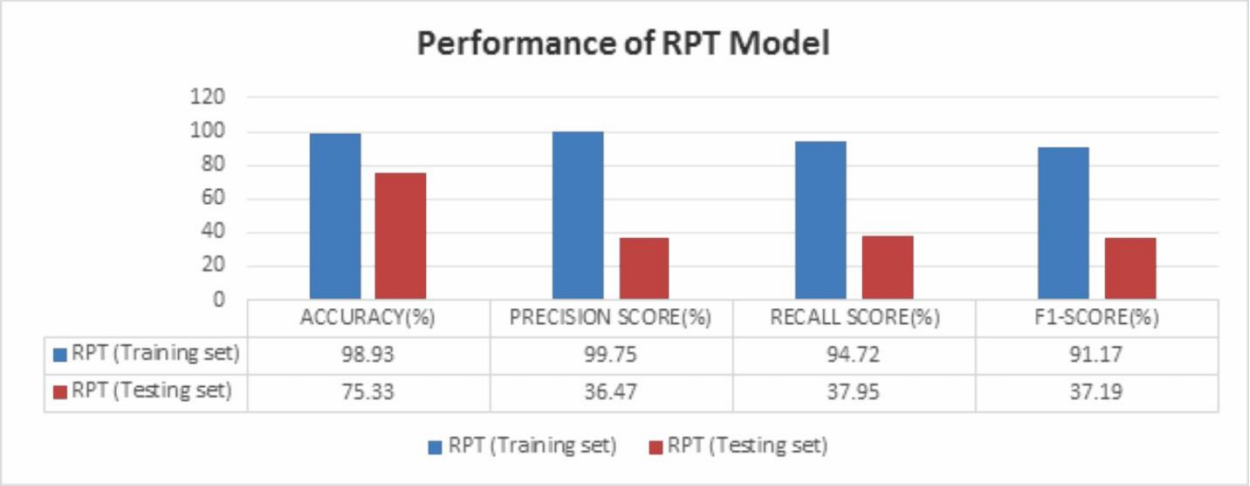


Figure 7: Performance of RPT Model with PSO

Figure 7 illustrates the performance of the RPT model with training and testing data. The accuracy, precision score, recall and F1-score of the RPT model for training data are 98.93%, 99.75%, 94.72% and 91.17% respectively.

Performance of Random Forest Model with PSO

Table 4: Performance evaluation of Random Forest Model with PSO

Models	ACCURACY(%)	PRECISION (%)	RECALL (%)	F1-SCORE(%)
RFC (Training set)	98.92	99.26	95.14	97.15
RFC (Testing set)	81.17	52.49	22.67	31.67

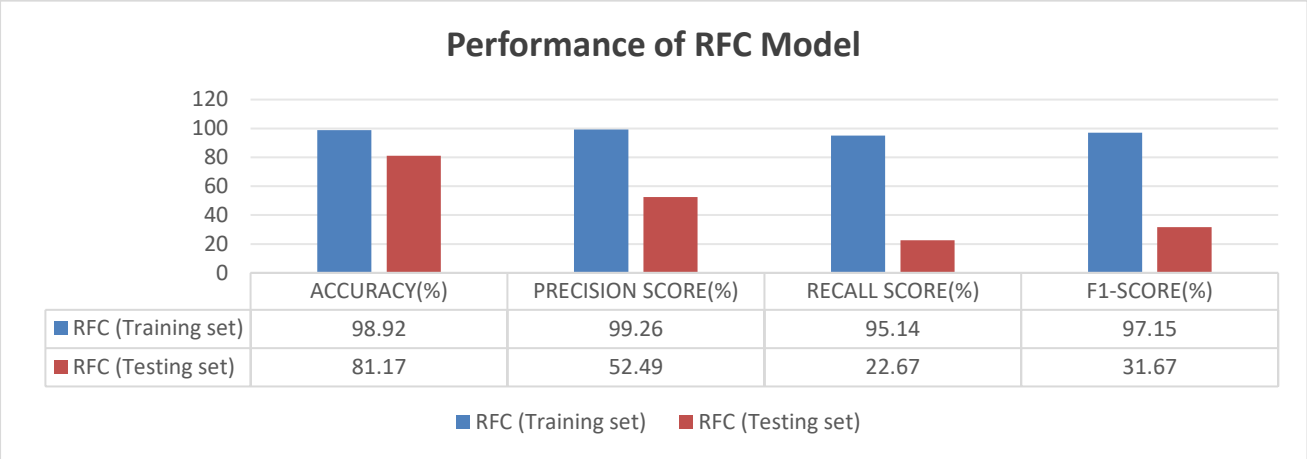


Figure 8: Performance of RFC Model with PSO

Figure 8. Illustrates the performance of the RFC model with training and testing data. The accuracy, precision score, recall and F1-score of the RFC model for training data are 98.92%, 99.26%, 95.14% and 97.15% respectively.

Performance of Artificial Neural Network Model with PSO

Table 5: Performance Evaluation of ANN Model with PSO

The performance of the ANN Model is shown in Table 5 and the graph representation is given in Figure 9.

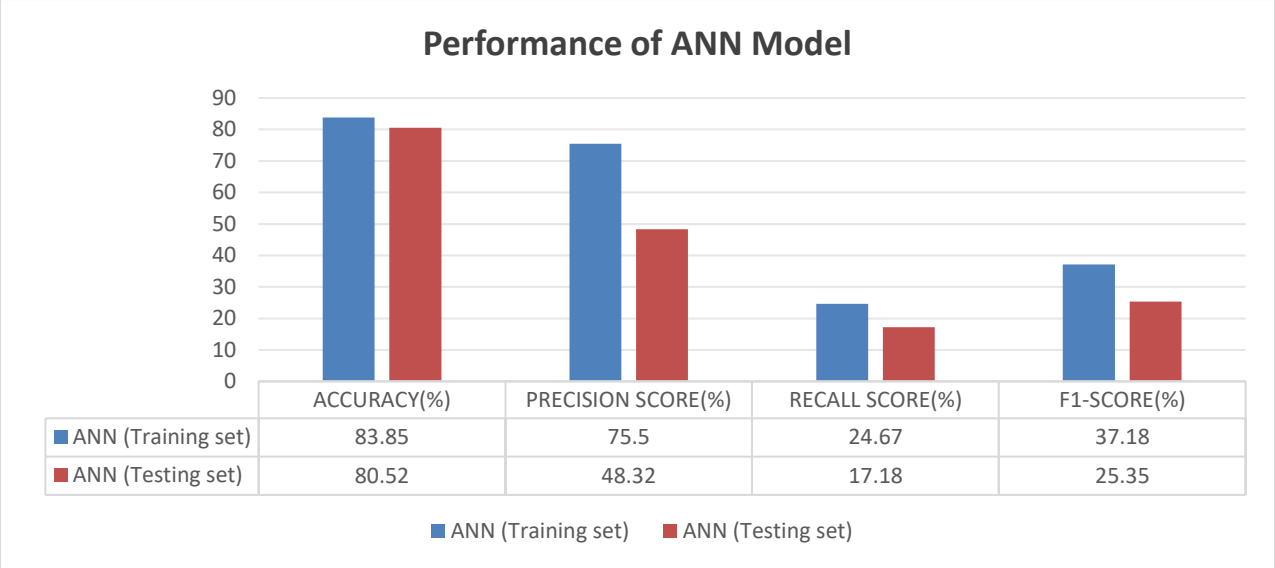


Figure 9 illustrates the performance of the ANN model with training and testing data. The accuracy, precision score, recall and F1-score of the ANN model for training data are 83.85%, 75.50%, 24.67% and 37.18% respectively.

Comparative Analysis of all the Models

The comparative analysis of learning techniques is compared as shown in Table 6 and the graphical representation is displayed in Figure 10.

Table 6: Analysis of SVM, RPT, RFC and ANN Models with PSO

Models	ACCURACY(%)	PRECISION (%)	RECALL (%)	F1-SCORE(%)
SVM (Training set)	82.60	83.86	12.63	21.95
RPT (Training set)	98.93	99.75	94.72	91.17
RFC (Training set)	98.92	99.26	95.14	97.15
ANN (Training set)	83.85	75.50	24.67	37.18

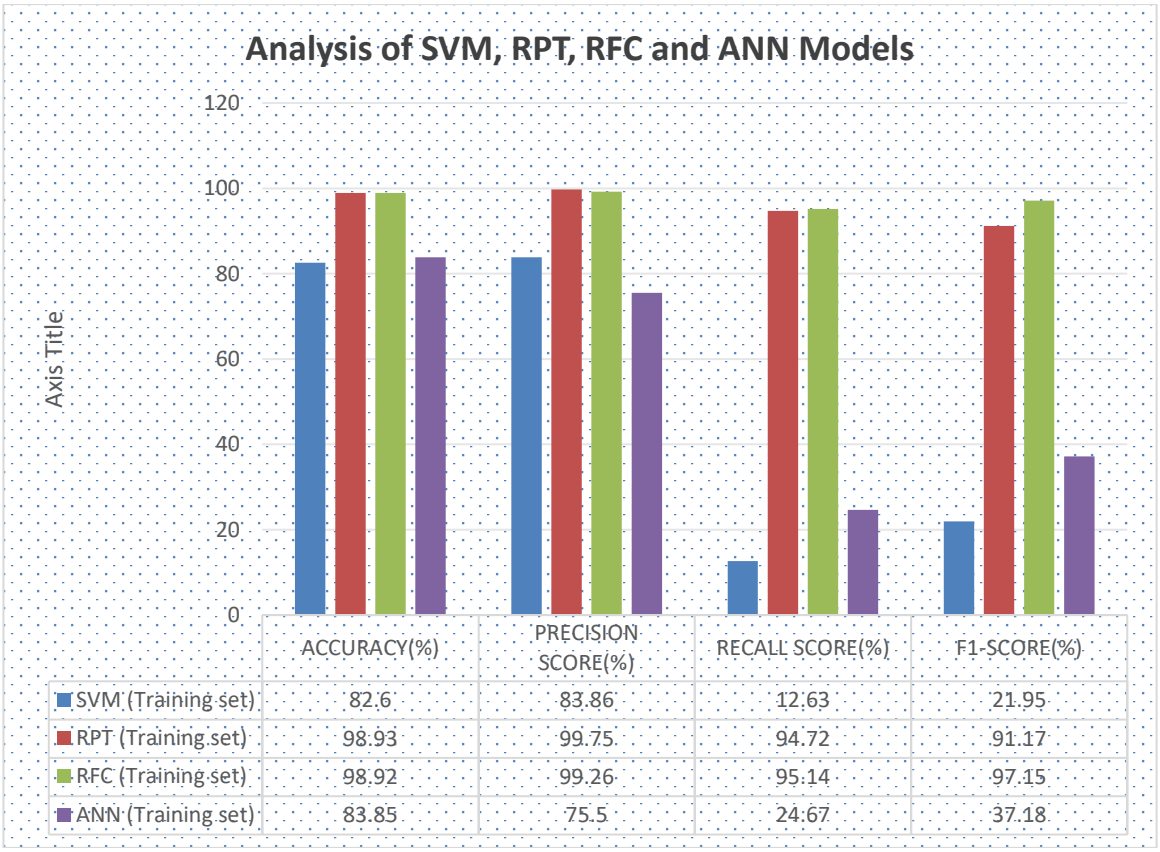


Figure 10: Analysis of all Models with PSO

Figure 10 illustrates how the RPT model performed better than the other models. The precision score, recall, F1-score, and accuracy of the RPT model are 99.75%, 94.72%, and 98.93%, respectively. According to the results of the performance measurements acquired, the RPT model outperformed the RFC model.

Performance Evaluation with the Existing Work

The developed model was compared with existing work as shown in Table 7.

Table 7: Comparison with existing work

Research paper	Authors and Year	Methodology	Results
Research on Software Defect Prediction Model Based on Complex Network and Graph Neural Network	Mengtian, Songlin, Yue, & Xu, (2022)	Complex Network and Graph Neural Network	Accuracy= 86.6% F-Measure=85.8% Matthews correlation coefficient=73.5%
examined Software Defects Prediction using Machine Learning Algorithms.	Madadi, Aravind and Kamal (2023)	ANNs (artificial neural networks), RF (random forest), RT (random tree), DT (decision table), LR (linear regression), GP	RPT Accuracy = 74.24 %, Precision=72%%, Recall=79%, F1-Score=75%, %

		(gaussian processes), SMOreg, and M5P.	
Developed Model			RPT Accuracy= 98.93%
	2024	SVM, RPT, RFC & ANN	Precision=99.75%
			Recall = 94.72%
			F1-Score=91.17%

The NASA database was employed as the standard database and the result has shown that the developed model outperformed the existing work in terms of Accuracy, precision, Recall and F1-Score.

Conclusion

In this research, ML techniques are utilized with feature selection techniques for software defect prediction. This research examined various well-known ML techniques on a freely available dataset to improve the accuracy of the dataset in comparison with previous research. All ML models are trained and tested through Python programming language using Jupyter Notebook. The analysis aimed to improve the accuracy performance of ML on the NASA dataset. Using NASA datasets, the effectiveness of several well-known machine learning classifiers, including Support Vector Machine (SVM), Random Forest, Recursive Partitioning Trees (Rep.Tree) and Artificial Neural Networks (ANN), was assessed for software defect prediction. The results showed that the RPT model performed better than all other models in terms of accuracy, precision score, recall, and F1 score. The RPT Classifier's accuracy, precision, recall, and F1-Score with the NASA dataset are 98.93%, 99.75%, 94.72%, and 91.17% respectively. Since RPT outperforms several other models in terms of classification, it is advised to utilize it in conjunction with PSO techniques for software fault prediction. To better the research in the future, scientists can additionally focus on the following. enhance model training, particularly when working with unbalanced datasets, devise techniques for augmenting defect data.

References

Akbar, A. M., Herteno, R., Saputro, S. W., Faisal, M. R., & Nugroho, R. A. (2024). Optimizing software defect prediction models: Integrating hybrid grey wolf and particle swarm optimization for enhanced feature selection with popular gradient boosting Algorithm. *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, 6(2), 169-181.

Angga, K. O., & Elish, M. O. (2008). Predicting defect-prone software modules using support vector machines*. Journal of Systems and Software,* 81(5), 649-660*.*

Alaa, H., Huang, S., Wu, Y., Hui, Z., & Zheng, C. (2019). A new weighted naive Bayes method based on information diffusion for software defect prediction. *Journal of Software Quality* 27(3), 923-968.

Ali, M., Mazhar, T., Al-Rasheed, A., Shahzad, T., Ghadi, Y. Y., & Khan, M. A. (2024). Enhancing software defect prediction: A framework with improved feature selection and ensemble machine learning. PeerJ Computer Science, 10, e1860.

Anjali, C., Punitha Malar Dhas, J., & Amar Pratap Singh, J. (2023). Automated program and software defect root cause analysis using machine learning techniques. *Journal of e-Informatics Software Engineering,* 64(4), 878-885.

Aimen, K., Gran, B., Nasir, A., Muhammad, S., & Mohamed, G. (2023). Software defect prediction analysis using machine learning techniques. Sustainability, 2, 15, 5517. https://doi.org/10.3390/su15065517

Ceylan, E., Kutlubay, F. O., & Bener, A. B. (2016). Software defect identification using machine learning techniques. *In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, 240-247.

Hammad, M., Alqaddoumi, A., & Al-Obaidy, H. (2019). Predicting software faults based on k-nearest neighbors classification*. International Journal of Computing and Digital Systems,* 8(5), 462-467.

Jyothi k., Aravind, E., & Kamal, M.V. (2023). Predicting faults in high assurance software. in *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, 2010, pp. 26–34. doi: 10.1109/HASE.2010.29.

Karedla, C., & Satyanada, R. (2023) Software Defect Prediction using Machine Learning Algorithms. Journal of emerging technologies and innovative research. *An international Scholarly Open Access, Peer Reviewed Refered Journal*. www.jetir.org(ISSN-2349-5162).

Madadi, N., Aravind, Y., & Kamal, D. (2023). A feature selection framework for software defect prediction. *In 2023 IEEE 38th annual computer software and applications conference,* pp. 426-435.

Prabha, C. L., & Shivakumar, N. (2020, June). Software defect prediction using machine learning techniques. *In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)* pp. 728-733.