Cluster Computing (2021) 24:2581–2595
https://doi.org/10.1007/s10586-021-03282-8

Principal component based support vector machine (PC-SVM): a hybrid technique for software defect detection

Mohd. Mustaqeem[1] · Mohd. Saqib[2]

Received: 14 November 2020 / Revised: 11 March 2021 / Accepted: 29 March 2021 / Published online: 16 April 2021

Abstract

Defects are the major problems in the current situation and predicting them is also a difficult task. Researchers and scientists have developed many software defects prediction techniques to overcome this very helpful issue. But to some extend there is a need for an algorithm/method to predict defects with more accuracy, reduce time and space complexities. All the previous research conducted on the data without feature reduction lead to the curse of dimensionality. We brought up a machine learning hybrid approach by combining Principal component Analysis (PCA) and Support vector machines (SVM) to overcome the ongoing problem. We have employed PROMISE (CM1: 344 observations, KC1: 2109 observations) data from the directory of NASA to conduct our research. We split the dataset into training (CM1: 240 observations, KC1: 1476 observations) dataset and testing (CM1: 104 observations, KC1: 633 observations) datasets. Using PCA, we find the principal components for feature optimization which reduce the time complexity. Then, we applied SVM for classification due to very native qualities over traditional and conventional methods. We also employed the GridSearchCV method for hyperparameter tuning. In the proposed hybrid model we have found better accuracy (CM1: 95.2%, KC1: 86.6%) than other methods. The proposed model also presents higher evaluation in the terms of other criteria. As a limitation, the only problem with SVM is there is no probabilistic explanation for classification which may very rigid towards classifications. In the future, some other method may also introduce which can overcome this limitation and keep a soft probabilistic based margin for classification on the optimal hyperplane.

Keywords Software defects detection · PCA · SVM · Feature optimization · Classification · PROMISE dataset

1 Introduction

Software plays an important role to make the world a global village. Nowadays almost every business, multinational companies, e-commercial industry, social networking, general-purpose, personal used, software for medical instruments and other organization running on software's platform which is increasing very quickly with time [1]. And the current situation of the world is horrible due to the pandemic Covid-19, there is a situation of lockdown throughout the world people are working from home, many educational institutes, businesses, are running through software it also creates a huge jump in software production industries [2]. So, looking at these entire situations proper working of the software is necessary but in-between these things, somewhere down the line the quality of the software is compromising which is a major challenge to the researchers and developers. Software developed for medical instruments faces many problems reported from software faults [3]. There are various techniques to solve the issues of the software. Software testing is one of them which is a part of the software development life cycle [4]. In software testing, a testing team is allocated to test the number of cases, compare the results to the expected performance, test quality and finds the bugs if present in the software this is manual testing which takes cost time and overall costly [5]. On daily bases with the friction of

✉ Mohd. Saqib
msaqib.cs@gmail.com

Mohd. Mustaqeem
mohdmustaqeem34@gmail.com

[1] CSE Department, Institute of Technology & Management (A.K.T.U), Aligarh, U.P, India

[2] Mathematic and Computing Department, Indian Institute of Technology (ISM), Dhanbad, Jharkhand, India

seconds, a huge amount of data is generating from different sources and new techniques, programming languages are introducing in the market to handle all these there is a requirement of reliable software system which must be robust, free from bugs and any defects, this is the gap among high quality software and lower one. To resolve current software problems, software testing is not enough, to find the defects in the modern-day software there is a requirement of automated tools, new techniques like data mining and machine learning approach will be used which predict the software defects in early stage because finding a little defects after completion of software may leads to huge loss of money, time and efforts. If software based industry founds any software defects in working time so it may collapse the system, degrades the industry valuation and reputation. For better project development, an automated method using machine learning is required whose focus should be on the quality of software, predict the software defects earlier so that the testing team may resolve them easily, improve the performance, reduced other hidden costs and time of the recipient.

The current using methods of software defects predictions have many limitations according to Paramshetti and Phalke [6] they have said that the ANN technique when working with NASA datasets cannot manage imprecise information but PC can work with it, we are working on small dataset so we have chosen SVM but clustering technique is also work on small datasets it has a limitation that the datasets should be unlabelled. Another algorithm, the Association rule is worked with the linear value of datasets but SVM can work with the both linear and non leaner value of datasets. SVM with different kernel function it provides better results for prediction. According to Thamilselvana and Sathiaseelan [7] the AdaBoost algorithm is affected by noisy data and outliers, it is also not robust of predicting the software defects while SVM is not affected by outliers, along with this cart algorithms has poor modelling with linear structure, KNN has slow processing, high time complexity in classification, it is difficult to find an optimal solution as shown below in Tables 1 and 2.

Keeping in mind from the above limitations of different algorithms we have proposed the PC-SVM model suitable for a given dataset that has a higher rate of defect prediction.

The manuscript has organised as follows: the second section named "Literature Review" containing all the related work followed by the third section "Modelling" in which we have to describe data collection, model overview and evaluation criteria for the model. The fourth section, "Implementation", is about the program and coding used to implement the proposed hybrid model. The fifth section "Result and Discussion" is containing the calculation and findings of the model followed by the conclusion as "Conclusion".

2 Literature review

To sustain community and business activities from Information systems there must a need of software on a huge level, to develop such software, industries use new technologies, but they get minute considerable growth over a long time. Hence, as the demand is increasing automation process is also coming in trend with novel software development technologies that can overcome the issues of production.

In the current situation, people are using computer software, almost everything in the world like corporate sector, transportation and telecommunication, financial activities, banking system, educational institutes and even the individual activities of social elements through social networking is all depends on software system where the information is sharing [8].

In recent years, there was a vivid rise in the demand for the software system. After the 1990s the internet came into use for the general public [9] use of software systems before that were very limited like government offices, Finance Corporation and Telecommunication companies

Table 1 Comparative analysis PC with other feature optimization approach

S. No	Algorithm	Drawback
1	Genetic algorithm	Genetic Algorithm has higher-level complexity and it is not worth in our dataset that's why we have used PC for faster and better execution it possesses built-in feature selection [34]
		Genetic Algorithm requires a large dataset but our dataset is small for which we require a versatile and easy to implement PC technique [34]
2	Correlation threshold	While selecting the correlation threshold manually is a very risky task, because it may drop the important features and useful information, for this in our dataset we have used the build-in feature selection technique PC. [34]
		Redundant features may occur but using a PC can eliminate them. [34]

Table 2 Comparative analysis SVM with other classification approach

S. No	Algorithm	Drawback
1	Decision tree classification	This classification technique performed poorly on small datasets and in our proposed model we have worked on small datasets on which SVM is better and effective [35]
		It is affected by the overfitting of datasets but SVM is free from the overfitting problem [35]
2	Random forest classification	The random Forest Classification algorithm is also affected by the overfitting of datasets and SVM is not sensitive to overfitting [35]

and which type of software system will use was also not know and now, from the last few decades, there is an enormous demand of software system they became an essential thing this transition is due to the development of high-quality software running on technically strong hardware machines. The alteration came with time in software development for smartphones i.e. mobile software [10].

Different types of software have been developed and running on multiple operating systems, for example, MAC, UNIX and Windows open source along with the gaming software for mobile phones. To sustain these types of activities software is an essential component. If we compare with hardware development, software development starts progress but their progress is slow the evolution achieved software technology which includes the growth of programming languages [11]. In this rising community new technology and tools are developing especially in the field of computer software, the huge amount of data is generating from the internet endlessly every second. As we are living in the information age and there is a huge impact on software development industries, software products of different companies are used in various fields of technology. Due to these conditions, researchers and developers need to maintain the quality, stability and reliability of software products and which may be solved by software testing [12]. Sneha and Malle said that software Testing may be defined as a method by which we can find the bug and removes them final output will be free from any kind of defects, if we talk about the quality of any software it will come when we test the software because the advancement in the field of computer science before starting the production of any software it goes to testing [13]. According to the American Journal of Engineering Research (AJER) research paper, the automation of the tests may be defined as the defects or bugs that come due to changing of code, can be identified easily if it is not detected and miss from any cause, a new test will be written so that it will detect in future [14]. If we use automation testing rather than the manual testing method the results will be different its objective is to reduce the number of test cases [15]. Automation in software testing is one of the methods to reduce time consumption, manpower and it also finds the hidden errors present in the system which reduces the overall cost of the system [16]. Now a day almost every software is tested automatically, to optimized software testing efficiency genetic algorithm is used [17]. For reducing the application testing moment and find out errors easily, one of the regression testing algorithms is used called bee colony optimization algorithm (BCO) [18] but BCO has the drawback of premature convergence but our proposed model(PC-SVM) is comparatively faster. Some tools are, Test Complete (TC), Quick Test Pro (QTP). The main objectives of these two tools are to minimize the resources in the code maintenance and improve effectiveness for code reuse [19] and it has various limitations like TC does not provide searching for internal relationships, it is not performed well with the large datasets but our model works with more efficiency. Software Test works(STW) tool is used for the integration of automated software testing into the production process [20]. All these tools and techniques are used testing the software to find errors, bugs or faults if present, to improve the performance of the system but still they are not able to detect numerous defects resulting down in the performance of software testing. It takes more time, the effort for testing but our model takes less time in comparison with STW. Furthermore, there is diversity among software and their matrices, complex software is developed. Object-oriented design metrics have been developed which may provide some important sources distribution and position of the defect so the accurate prediction may save the costs in the testing process [21]. Previously, many research has employed many algorithms to do software defects detection which is as follows artificial neural networks (ANN), deep learning (DL), linear regression, K-nearest neighbors (KNN), decision tree, SVM, etc. [22]. In [23], the authors have proposed many statistical and mathematical approaches to detect software faults. Gondra says in [24] to measure the performance for excessive and indecision datasets a machine learning using ANN technique is developed, in which to train ANN, historical data of software metric values with several errors has given, to measure performance, sensitivity analysis is

performed and ANN is trained using the training data and different evaluation criteria but ANN has a limitation of overfitting the data and it needs large datasets for processing but our proposed model is well suited with the small set of data and SVM uses L2 regularization to minimize the overfitting problem. One of the techniques of Software defects prediction SVM is used in many areas to predict defects but their rate of prediction is limited due to accuracy view. To enhance accuracy feature selection and optimization techniques are used P-SVM which is the hybrid of Particle Swarm Optimization (PSO) and SVM algorithm is used for optimization [25]. This technique is to improve the more informative feature selection technique with reduced dimensions that can be obtained by text documents clustering using the PSO algorithm [25]. Text clustering is a technique in which a text can be classified into many groups which degrade the performance and increase the computational time. So, for the selection of more informative features, a hybrid technique of particle swarm optimization and genetic operators is used [26]. But there is a limitation of using PSO it easily falls into local optimisation in high-dimensional space it also has a low convergence rate in the iterative processing it works for large datasets but our hybrid model PC-SVM will work more effective rather than the P-SVM hybrid model. According to L. Abualigah group search optimizer (GSO) algorithm [27] is helping to solve multiple optimization problems it uses animal finding pattern to solve a problem. Moreover, a meta-heuristic algorithm optimizer is also be used to a solved optimization problem in many fields and a Multi-verse optimizer algorithm (MOA) is used to tell many of the applications of various optimization techniques [28] and also MOA has a low convergence rate but SVM manages the high dimensionality and gives effective results even with the small data. As we have shown the below mentioned Table 3 for the limitations of multiple algorithms.

Apart from this, in medical level IoT devices the sensitive data are communicated through an unsecured network. Various ML algorithms are using to block the cyberattack as Swarna Priya [29] has used the DNN hybrid model block cybersecurity attacks and intrusion detection in medical-related Internet of Medical Things (IoMT) devices but DNN has limitations of low performance with the noisy data and it requires a high amount of data that works on the large set of data while SVM it is free from the effect of outliers. According to Bhardwaj et al. [30] for blocking cybersecurity attack, the smart contracts decentralized applications the penetrating testing framework is used which can reduce the attacks. Bhattacharya [31] has also said in their manuscript that intrusion detection can be done by a hybrid of PCA-Firefly algorithm that uses for dimensionality reduction for network security. From the above-mentioned algorithms the hybridization with PCA can result more efficient.

According to Prabha and Shivakumar [32], the bug detection can be improved through various hybrid algorithm like the random forest, naïve Bayes and SVM but the author has worked on reducing the time complexity and robustness of the model which we have done with greater efficiency in our model along with overcoming the curse of dimensionality problem. In our paper, we are presenting the novel approach by combining the two most promising algorithm for optimization and feature selection to obtain better results with more accuracy. We are using a hybrid of SVM and PCA algorithm whose results and accuracy are more than the below-mentioned algorithm in our knowledge.

3 Modeling

In this section, we have explained dataset detail and also the methodology behind the hybridisation of two models which are the following:

Table 3 Comparison of algorithms shortcomings with PC-SVM model

S. No	Algorithms	Shortcoming
1	AdaBoost	This algorithm is affected by outliers and not effective in predicting the errors but SVM is free from outliers and effective in predicting the software defects [7]
2	CART	It has poor modeling with linear data while SVM can work with both linear as well as non-linear datasets [7]
3	KNN	It has a high rate of classification in comparison with SVM [7]
4	Neural Network	It works good with large datasets and taking time in training the datasets while our model has small datasets for that PC-SVM is better [7]
5	Chao Genetic	It has difficulty in providing the optimal solution while PC can provide [7]
6	EM model	It also does not guarantee in providing the optimal solution but our model PC-SVM can give [7]

3.1 Data description

The data has been collected from the PROMISE datasets repository [29]. We have used CM1 and KC1 datasets to implement the proposed model. In Table 4, we have given descriptions of various attributes used for analysis. Furthermore, we have split both the datasets into two parts: one more training the model and the second is for testing the data. In CM1 data 240 observations used for training the model and 104 observations for testing the model. Same as in KC1, 1476 observations have used for training and 633 records have employed for testing the model.

3.1.1 CM1 dataset detail

In CM1 Dataset, there are 344 observations and we split 240 observations for training and 104 observations for testing purpose. After applying PCA we have found 5 attributes are covered 99% confidence interval.

3.1.2 KC1 dataset detail

In KC1 Dataset, there are 2109 observations and we split 1476 observations for training and 633 observations for testing purpose. After applying PCA we have found 7 attributes are covered 99% confidence interval.

3.2 Hyperparameters

3.2.1 Hyperparameters of the model

There are three main hyperparameters (C, Gamma, and kernel) used in the fitting proposed model which are and their effects described below:

Kernel Kernel is defining a hyperplane function type. There are many kernels available which used to implement SVM according to the training dataset we used e.g. linear, non-linear, polynomial, Gaussian kernel, Radial basis function (RBF), sigmoid etc. For the proposed model, we have found RBF is the best-suited kernel. RBF is a widely used kernel function because it has a robust classification strategy and can be used without any prior knowledge of the dataset. It can be represented by the following equation:

$$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

where $\gamma > 0$ and $\gamma = {}^{1}/_{2\sigma^2}$.

C SVM is all about creating decision boundaries using kernels we described or chosen previously as shown above in Fig. 1. Suppose, if we take a very tight boundary and classify it over a single fitted line then a single small noise may lead to misclassification. It's nothing but an overfitted model which may perform extremely well with training data but not with testing data. To overcome this problem concept of soft margin came into existence and it can be

Table 4 Attributes description

Attribute name	Description of attribute
LOC	Counts the total number of line in the module
Iv(g)	Design complexity analysis (McCabe)
Ev(g)	McCabe essential complexity
N	Number of operators present in the software module
v(g)	Cyclomatic complexity measurement (McCabe)
D	Measurement difficulty
B	Estimation of effort
L	Program length
V	Volume
I	Intelligence in measurement
E	Measurement effort
Locomment	Line of comments in software module
Loblank	Total number of blank lines in the module
uniq_op	Total number of unique operators
uniq_opnd	Total number of unique operand
T	Time estimator
Branchcount	Total number of branch in the software module
total_op	Total number of operators
Total_opnd	Total number of operators
Locodeandcomment	Total number of line of code and comments
Defects/Problems	Information regarding defect whether the defect is present or not

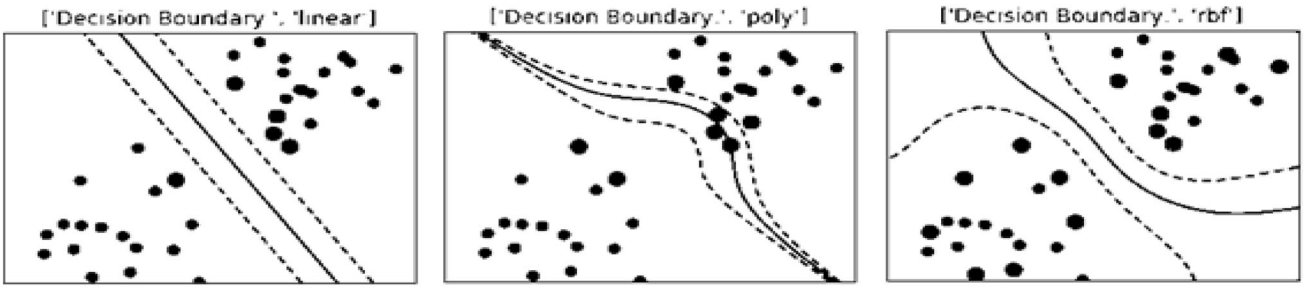

Fig. 1 Decision boundary

achieved by optimizing the value of c parameter. This parameter is nothing but a penalty for the miss classification which may belong to any positive real number ($c \in \mathbb{R}^+$). For large c, a penalty will be high and SVM will reduce the misclassification and small value it will be less and flexible with classification Fig. 2. The penalty is not the same for all misclassified points or observation; it's directly proportional to the distance from the boundary kernel draws. For the proposed model we have taken c = 0.1 which may obtain from GridSearchCV.

Gamma Gamma is a hyperparameter that decides the non-linearity of the RBF Kernel. Low value of gamma will give a linear boundary which may lead to underfitting and a high value of the gamma will be a very tight boundary and do very brutal classification which may lead to overfitting as shown below Fig. 3. For, linearly separable data C Parameter tuning is enough but for the proposed (non-linear problem) C parameter and Gamma both need to be tune simultaneously. For tuning, we employed GridSearchCV and the value of Gamma belong to $0.0001 < \text{gamma} < 10$.

3.2.2 Hyperparameter tuning method

There are various hyperparameter tuning method has already been proposed and we employed GridSearchCV. The logic behind GridSearchCV is simple to create a grid and make all the possible combinations of hyperparameters. Now, compute all and find the optimum one. For, all this we can simply use the Scikit-Learn GridSearchCV library and it will return the optimum value as follows:

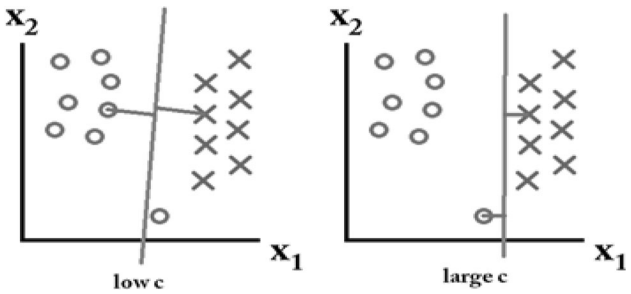

Fig. 2 Effects of C parameter on SVM model fitting

3.3 Implementation setup and working environment

The computer code, written in Python.3, debugged on SPYDER which can be download from the link: https://www.spyder-ide.org/. SPYDER is a non-profit organization created to develop open-source software, open-standards, and services for interactive computing python. The code can be access from the public GitHub repository: https://github.com/mmustaqeem/software-defects/tree/main/Code_PCA_SVM. In the repository two separate folders are present; for the CM1, and KC1 Experiment. In both, the folder executable file (.py) added to the review of code. The database files (.csv), used in the code, also attached in the same folder.

3.4 Proposed model

The proposed model is the combination of the following two models PCA and SVM. The SVM is strong enough to classified defects and non-defective software observations.

```
opti_param =
{'C': [0.1, 1, 10, 100, 1000],'gamma': [1, 0.1, 0.01, 0.001, 0.0001],'kernel': ['rbf']}
grid =GridSearchCV(SVM(),opti_param)
grid.fit(X_train, y_train)
```

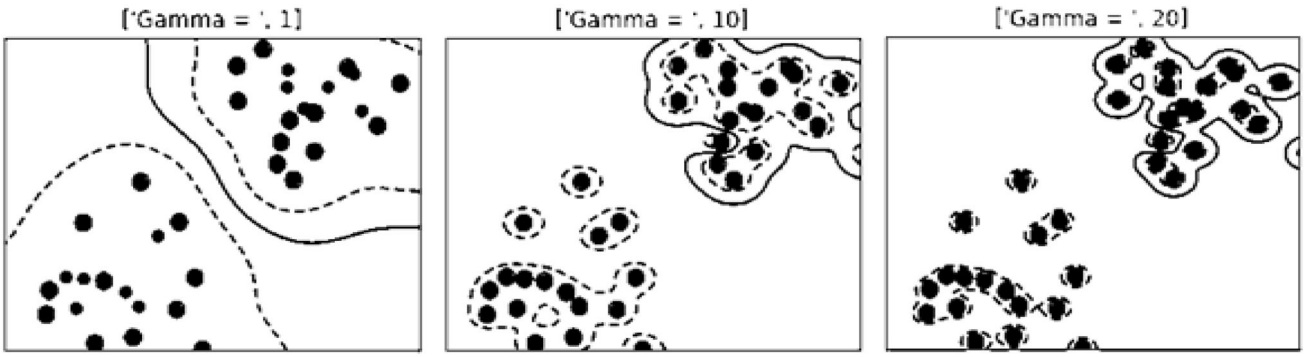

Fig. 3 Gamma hyperparameter

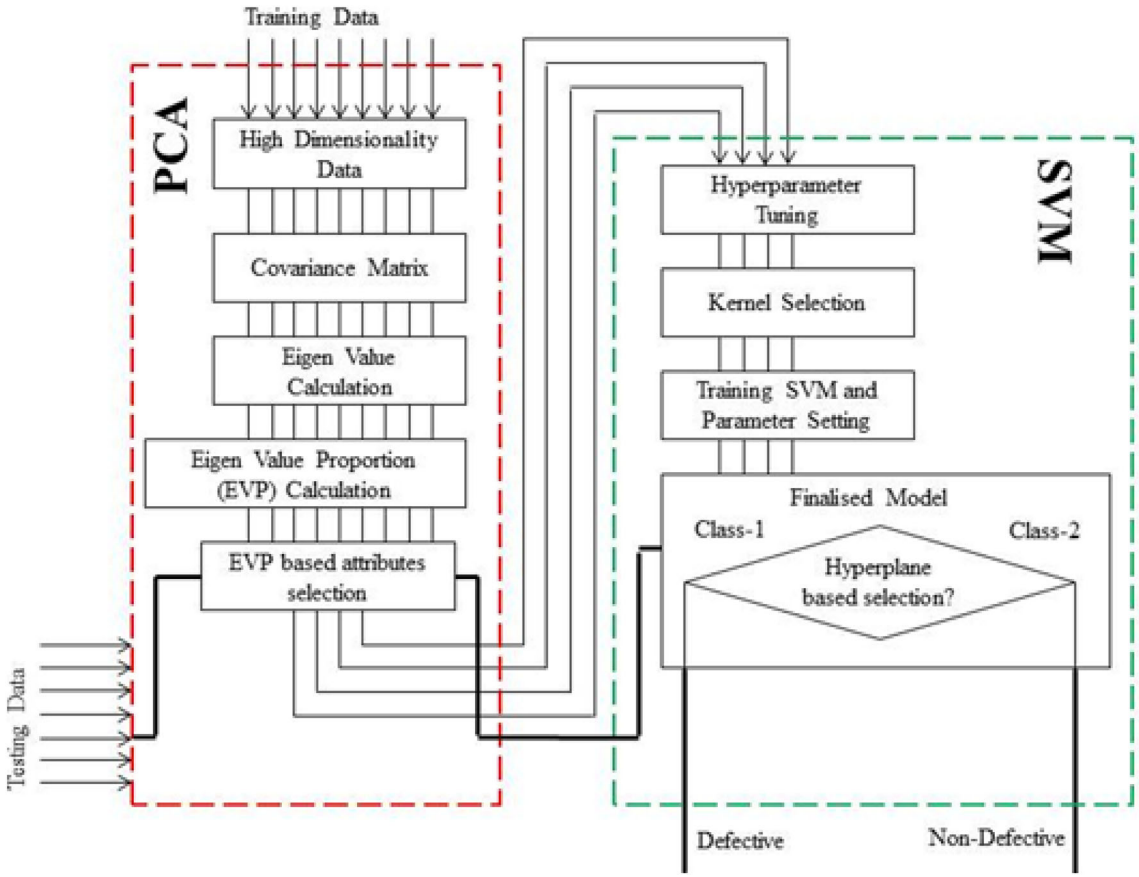

Fig. 4 PC-SVM model representation

But, we have also included PCA to reduce the time complexity and robustness of the analysis. PROMISE data repository datasets like KC1and MC1are full of various attributes which required high processing power. Therefore, the proposed reduce robustness. Figure 4 explains clearly how high dimensionality data goes inside the PCA (red colour dotted rectangle part). Then, low dimension data are given to SVM to do classification (green colour dotted rectangle part). After all, parameter tuning completed and the model becomes to do classification prediction. Now, we passed high dimension data to PCA and then low dimension data (output of PCA) transfer to the SVM model to classify fitted optimal hyperplane.

3.4.1 Principal component analysis

In this classification problem, finding necessary features is one of the major tasks of machine learning, as the time passing new development in technology is continuously going on, which makes this task a little bit easier, like when

we combined our task with AI, it tries to solve these problems in an easier, effective and efficient way, it reduces the time for getting results. AI applies a huge amount of data from that we have to find the defective datasets, working with large datasets contain many numbers of features, and if it's quantity become equal or larger than the number of observations stored in datasets due to which machine learning model faces overfitting problem. To avoid these types of problems, dimensionality reduction technique is applied and there are various techniques for features extraction and variable reduction in machine learning, which is PCA, is an unsupervised machine learning algorithm and multivariate statistical technique which reduces the number of variables by extracting important information from the data; it also affects the data matrix, to aspire maintaining as much information as possible and extracts the main pattern from the data. L. J. Williams said in their paper[26] that, this technique binds highly inter-correlated variable together to form a smaller number of a simulated set of new orthogonal variables called 'Principal Components' that tells variance in the data and to check the quality of the PCA model, cross-validation techniques are applied like bootstrap and jack-knife. Mathematically, the Principal component problem can be solved in the following steps;

$$\begin{cases} Z_1 = \overrightarrow{\omega_1}^T.\overrightarrow{P} = \omega_{11}P_1 + \omega_{12}P_{12} + \cdots + \omega_{1n}P_n \\ Z_2 = \overrightarrow{\omega_2}^T.\overrightarrow{P} = \omega_{21}P_1 + \omega_{22}P_2 + \cdots + \omega_{2n}P_n \\ \quad \ldots\ldots \\ Z_n = \overrightarrow{\omega_n}^T.\overrightarrow{P} = \omega_{n1}P_1 + \omega_{n2}P_2 + \cdots + \omega_{nn}P_n \end{cases} \quad (1)$$

In the above expression, P_i is the real variable, Z_i is the principal component and $\overrightarrow{\omega_i}$ is the coefficient vector.

The estimation of $\overrightarrow{\omega_i}$ can be done maximizing $Var(Z_i)$ with limited conditions of $\overrightarrow{\omega_1}^T.\overrightarrow{\omega_1} = 1\&$

$$Cov(Z_i, Z_j) = \overrightarrow{\omega_1}^T.\sum.\overrightarrow{\omega_1}^T = 0, j = 1,2,\ldots,i-1, where =$$

it provides the covariance matrix of $\overrightarrow{P}$.

Covariance matrix of $\overrightarrow{P} = (P_1, P_2, \ldots P_n)^T, = (\sigma_{ij})_{n\times n}$ is a symmetric non-negative definite matrix.

So, it has n features solutions$\lambda_1, \lambda_2, \ldots, \lambda_n$, and n features vectors. Suppose $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n \geq 0$ and the orthogonal unit eigenvectors are $\overrightarrow{e_1}, \overrightarrow{e_2}, \ldots, \overrightarrow{e_n}$. the *ith* principal component $P_1, P_2, \ldots P_n$ as given below.

$$\boldsymbol{Z_i} = e_{i1}\boldsymbol{P_1} + e_{i2}\boldsymbol{P_2} + \ldots + e_m\boldsymbol{P_n}, \boldsymbol{i} = 1, 2, \cdots, \boldsymbol{n} \quad (2)$$

$Var(Z_i) = \overrightarrow{e_1}^T.\sum.\overrightarrow{e_1}^T = \lambda_i\&$-
is first p principal components.

Combined rate can be given as:

$$ACR(c) = \sum_{i=1}^{c} \lambda_i / \sum_{i=1}^{n} \lambda_i \quad (3)$$

3.4.2 Support vector machine

As we know, a machine learning algorithm can be categorized in the term of adopted paradigm in three ways—Supervised, Unsupervised, and Semi-Supervised. Moreover, it can be categorized in the terms of types of problems it solves e.g. Regression, Classification, and Clustering. Our problem of Software defects prediction is a kind of classification problem. There are two types of data we have, one of them have software defects and another have no defects. Since the rise of Artificial Intelligence, several algorithms are proposed to achieve classification problem in machine learning like logistic regression, decision tree, random forest, etc. but the SVM [28] is the finest supervised machine learning paradigm having the following advantages-

- It uses L2 Regularization to prevent overfitting in the model,
- Give appropriate results even on small data,
- Various Kernel-Tricks to fit the complex function and relationships among features,
- Handel non-linearity of the data,
- Hyperplane dividing rule provides stability in the model,
- Manages high-dimensionality of the data.

Instead of minimizing the error of prediction, SVM has more focus on maximizing the decision boundaries of classification that is why the separation of classes, either + 1 (positive class) and -1 (negative class) takes place by the hyperplane. In the SVM, we can use high-

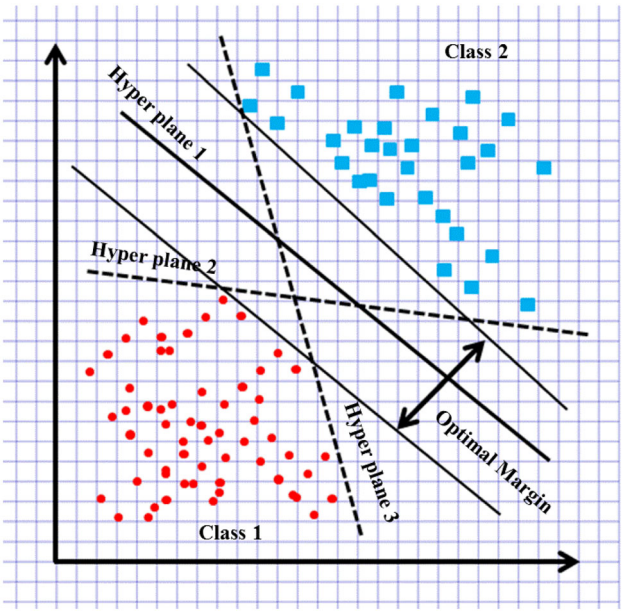

Fig. 5 2D graph hyperplane

dimensional (n dimensions) datasets as input that cannot be visualized. However, if we process a dataset having n = 2 (2D), it can be demonstrated on a 2D graph (Fig. 5) and a hyperplane will be a line that distinguished classes. Moreover, if data is n-dimensional the hyperplane will be an (n-1) vector function which can mathematically represent as follow

$$y = w_0x_0 + w_1x_1 + \ldots + w_{n-1}x_{n-1} + b \tag{4}$$

More generally, it also represents

$$y = W^TX + b \tag{5}$$

where W is weight vector, X is an input feature vector and b is bias. Once a hyperplane found, our hypothesis based on SVM can be formulated as below

$$f(y) = \begin{cases} Class\,1\ if\ y \leq 0 \\ Class\,2 if\ y\ >\ 0 \end{cases} \tag{6}$$

Many hyperplanes can be drawn by tuning W and b but the hyperplane with optimal margin will be selected. The optimal margin can define as the maximum possible perpendicular distance between the hyperplane and each class. For example, in Fig. 5, hyperplane 1 has an optimal margin from both, class 1 and class 2. The optimal margin is found by minimizing the cost function or objective function. The cost function can be defined as

$$J(W) = \frac{1}{2}\|W\|^2 + \frac{1}{n}\sum_{i=0}^{n}\max(0, (1 - y_i * (W^TX + b)) \tag{7}$$

Even the predictions are correct and data classified correctly by hypothesis, SMV used to add a penalty for all those y_i which are near to the boundaries $(0 < y_i < 1)$. Our objective is to minimize $J(W)$, in the term of optimal W so differentiating Eq. 7 concerning W we get the gradient of a cost function,

$$\nabla_W J(W) = \frac{\partial J(W)}{\partial W} = \frac{1}{n}\sum_{i=0}^{n}\begin{cases} W\ if\ \max(0, (1 - y_i * (W^TX + b)) \\ W - y_ix_i otherwise \end{cases} \tag{8}$$

As far as we have calculated $\nabla_W J(W)$, now we can update the weights (W) using Eq. 6

$$W_{new} = W_{old} - \alpha[J(W)] \tag{9}$$

We repeat the process till the smallest $J(W)$ discovered.

It is very often data may not be linearly separable that is why we need to plot a decision boundary between the classes instead of separating by a hyperplane. Now, to handle the non-linearity of the datasets, we need to transform Eq. 5 into a decision boundary.

$$y = W.\phi(X) + b \tag{10}$$

In Eq. 10, $\phi(X)$ is known as the kernel function. There are many kinds of kernel functions that are available to implement SVM like linear, polynomial, exponential, etc. but in the purposed model we are using Radial Basis Function (RBF)[28]. It is based on Euclidean Distance and parameter σ defines the smoothness of the boundaries[33].

$$\phi(x) = \exp\left(-\frac{\|x - \bar{x}\|^2}{2\sigma^2}\right) \tag{11}$$

where $\|x - \bar{x}\|^2$ square of Euclidean Distance between any single observation x and mean of the training sample $\bar{x}$.

Algorithm

Start

Step-1 Import Libraries // Import all the necessary libraries and file to implementation
Step-2 Data = read(File) // Read the available data
Step-3 Initialize weights W, and bias b with any arbitrary number
Step-4 Feature Optimization // Select the appropriate attributes for the prediction
Step-5 TrainDataSet, TestingDataSets = Data.Split (Ratio) //Splitting the data in two-part, one for training the model, another for testing the model in the suitable ratio
Step-6 Define y // Equation 5 or 10
Step-7 Define h(y) // Equation 6
Step-8 Define J(W) // Equation 7
Step-9 Calculate $\nabla_W J(W)$ //Equation 8
Step-10 Repeat for minimum J(W):
Update W // Little change in weights to found an optimal margin (Equation 9)
Call Steps 6-9 // Re-calculate the $\nabla_W J(W)$ by making prediction using updated W
End
Step-11 Calculate accuracy // To check the efficiency of the model

$$Accuracy \leftarrow \frac{Totalno.\ ofcorrectpredictions}{Totalno.ofpredictions} * 100$$

Step=12 Output // Giving the accuracy of the model as output

Stop

3.5 Evaluation criteria

To evaluate the results, we have employed a confusion matrix. The confusion matrix is a simple classification between predicted and actual values like Table 5 is a clear demonstration of the same. After getting the values, we calculate further metrics e.g. accuracy of the model, precision, F-score etc.

There are three confusion matrices created for each dataset. A confusion matrix is a representation of actual value and predictions based on testing datasets (Table 4). The accuracy of the model determines by the confusion matrix and some other performance parameters (Recall, Precision, etc.) also can be calculated.

The accuracy of the model can be calculated from the following formula-

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

The precision of the model can be calculated using the following formula

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

Recall of the model calculated using correctly predicted positive observations and total numbers of positive models in testing datasets

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

Finally, F-measure is computed from the following formula

$$F = \frac{2 * Precision * Recall}{Precision + Recall}$$

Table 5 Confusion matrix demonstration

Predicted value		
	Negative	Positive
Actual value		
Negative	True Positive (TN)	False Negative (FP)
Positive	False Positive (FN)	True Negative (TP)

		Predicted	
		Defective	Non-Defective
Actual	Defective	2	83
	Non-Defective	2	546

Fig. 6 Confusion matrix for KC1 dataset using a proposed hybrid classifier

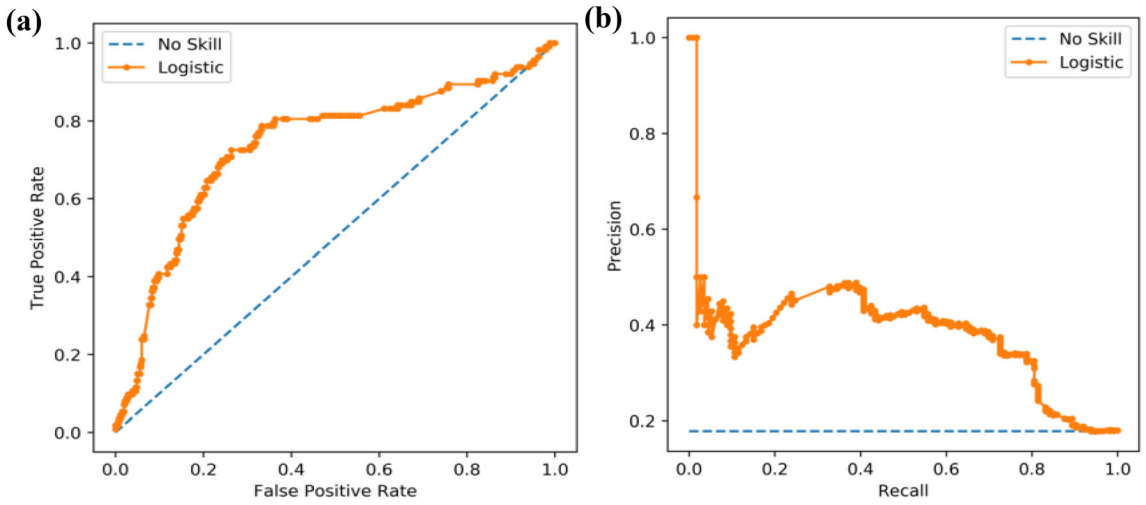

Fig. 7 **a** ROC curve analysis, **b** Precision–recall analysis for KC1 Dataset

		Predicted	
		Defective	Non-Defective
Actual	Defective	0	4
	Non-Defective	1	99

Fig. 8 Confusion matrix for CM1 dataset using a proposed hybrid classifier

Table 6 Statistical performance analysis for CM1 and KC1 datasets using PCA and SVM

Dataset	Precision	Recall	F-Measure	Accuracy
CM1	96.1	99.0	97.5	95.2
KC1	86.8	99.6	92.8	86.6

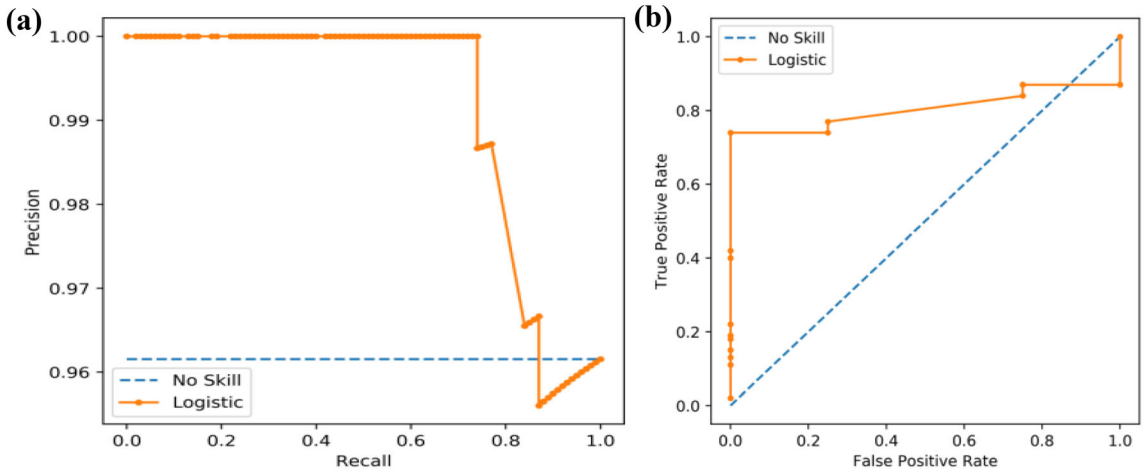

Fig. 9 **a** Precision and recall analysis for MC1 dataset, **b** ROC curve analysis

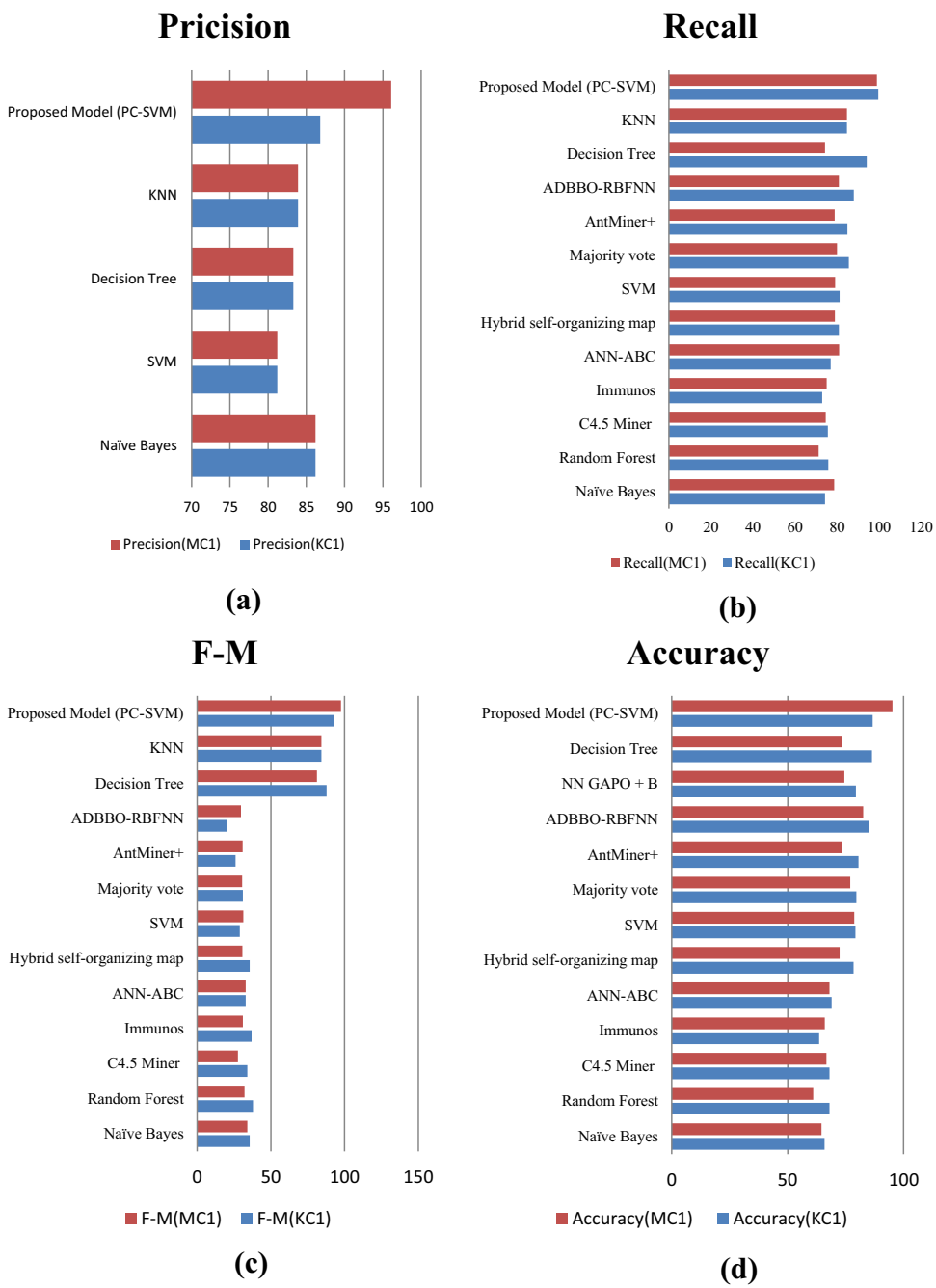

Fig. 10 **a** Precision for CM1 & KC1 dataset, **b** Recall analysis for CM1 & KC1 dataset, **c** F-M analysis for CM1 & KC1 dataset, **d** Accuracy analysis for CM1 & KC1 dataset

4 Implementation

For the implementation of the proposed model, we have written the code in Python language and to implement the program Spyder platform. We have implemented the proposed model using the following two datasets:

4.1 Using KC1 dataset

Figure 6, demonstrated the confusion matrix of KC1 datasets using the proposed model PC-SVM.

Figure 7a has shown the ROC curve of the implemented model and every point is above the no-skill line. Figure 7b has represented the Precision-Recall curve and the same as the ROC curve every point is above than the no-skill line.

4.2 Using CM1 Dataset

In Fig. 8 we have demonstrated the confusion metrics of the CM1 dataset.

Figure 9a demonstrated the Precision and recall curve and most of the value of precisions is above the no skill

Table 7 Performance comparison of the proposed model and previously proposed models

NASA dataset	Techniques	Precision	Recall	F-M	Accuracy
KC1	Naïve Bayes [30, 35]	86.2	74.33	35.71	65.87
	Random Forest [30]	–	75.89	37.91	67.99
	C4.5 Miner [30]	–	75.64	34.05	68.01
	Immunos[30]	–	72.91	36.92	63.55
	ANN-ABC [30]	–	77	33	69
	Hybrid self-organizing map [31]	–	80.94	35.67	78.43
	SVM [32, 35]	81.2	81.27	28.96	79.24
	Majority vote [32]	–	85.62	30.98	79.66
	AntMiner + [32]	–	84.99	26.11	80.51
	ADBBO-RBFNN [34]	–	87.95	20.24	84.96
	NN GAPO + B [36]	–	–	–	79.4
	Decision Tree[35]	83.3	94.1	87.78	86.35
	KNN[35]	83.9	84.7	84.3	–
	Proposed Model (PC-SVM)	86.8	99.6	92.8	86.6
CM1	Naïve Bayes [30, 35]	86.2	78.65	34.09	64.57
	Random Forest [30]	–	71.29	32.17	60.98
	C4.5 Miner [30]	–	74.66	27.68	66.71
	Immunos[30]	–	75.02	30.99	66.03
	ANN-ABC [30]	–	81	33	68
	Hybrid self-organizing map [31]	–	78.96	30.65	72.37
	SVM [32, 35]	81.2	79.08	31.27	78.69
	Majority vote [32]	–	80	30.46	77.01
	AntMiner + [32]	–	78.88	30.9	73.43
	ADBBO-RBFNN [34]	–	80.96	29.71	82.57
	NN GAPO + B [36]	–	–	–	74.4
	Decision Tree[35]	83.3	74.23	81.2	73.49
	KNN[35]	83.9	84.7	84.3	–
	Proposed Model (PC-SVM)	96.1	99.0	97.5	95.2

line. Same as in Fig. 9b ROC curve of the proposed model has been represented. Table 6 has shown the values of different metrics taken for evaluation of the proposed model.

5 Results and discussion

After the implementation of the purposed model, we compared our model with previous studies based on precision, recall, F-measure, and accuracy of classification. We have found that the proposed model is more accurate than the other. Apart from accuracy, the proposed model also used dimensionality reductions or feature deduction that reduce the computational time and overhead of analysing whole data.

In KC1 dataset analysis, we have found precision 86.8, recall 99.6, F-measure 92.8, and accuracy 86.6 that is a significant improvement in the analysis. Same as in CM1 dataset analysis, we have found precision 96.1, recall 99.0, F-measure 97.5, and accuracy 95.2. In the Fig. 10, improvement in the all the parameters of evaluation in both the datasets (CM1 and KC1) can be clearly visualize. Dark red colour has denoted the analysis of CM1datasets in all the four bar graphs (Fig. 10a for precision, (b) for recall, (c) for F-M score, and (d) for accuracy). And, similarly, blue colour represents the analysis of KC1 dataset. Bar graphs are high for every metrics for proposed model. Table 7 has a tabular representation of the comparative analysis of all other traditional and conventional methods.

6 Conclusion

Measurement of quality of any software mainly depends on, how efficiently and in early-stage its defects can be predicted. Detection of software bugs in an early stage can be done by various techniques, previously developed by researchers and scientists. Due to the learning mechanism of classifiers, the most capable approach considered is

machine learning-based. We have examined the performance of the proposed model which is the combination of PCA and SVM implemented on the PROMISE dataset repository by using previously developed conventional and traditional methods. We select PCA to overcome the problem of the curse of dimensionality and reduce the computational requirements of the proposed task. It was the major problem with the previous research methodology mentioned in Tables 1 and 2 then after we hybrid this approach with SVM, which is a very powerful methodology of classification in ML. We have found SVM more sustainable in terms of many cases like it can give better result with small datasets and doesn't effects by outliers. When a small culprit in the dataset may lead us to the wrong classification then SVM's this quality will be a boon for the model.

We have test results on the bases of F-measures, Recall, Accuracy and Precisions. We have found that PC-SVM has given accuracy 86.6% with 86.8% precision, 99.6% Recall and 92.8% F-measure on the KC1 dataset. Similarly, the CM1 dataset purposed model has given the accuracy of 95.2% with 96.1% precision, 99% recall and 97.5% F-measure. In future, using this approach will lower the maintenance cost of software and will help researchers to detect bugs in software in less time with high accuracy. It will open a new door for research; in the field of software detect prediction using the machine learning approach.

The major limitation of the proposed model is that there is no probabilistic explanation for the classification. As we know SVM classifies optimal hyperplane which is a very rigid approach in some cases. Therefore, in future, we may work on classification using SVM methodology along with probabilistic theorems e.g. Bayes Theorem which may give us a probability base selection strategy over hyperplanes. This kind of hybridisation of two different approaches may give us a soft-margin in SVM.

References

1. Florence, J.R.L., Arya, A.: A review on software defect prediction techniques using product metrics. Int. J. Database Theory Appl. **10**(1), 163–174 (2017)
2. del Rio-Chanona, R.M., Mealy, P., Pichler, A., Lafond, F., Farmer, D.: Supply and demand shocks in the COVID-19 pandemic: an industry and occupation perspective. Oxf. Rev. Econ. Policy **36**, 1–38 (2020)
3. Gewaltig, M.-O., Cannon, R.: Current practice in software development for computational neuroscience and how to improve it. PLoS Comput. Biol. **10**(1), e1003376 (2014)
4. Jamil, A., Arif, M., Abubakar, N., Ahmad, A.: Software testing techniques: a literature review (2016)
5. Sharma, R.M.: Quantitative analysis of automation and manual testing. Int. J. Eng. Innov. Technol. **4**(1), 252–257 (2014)
6. Cohen-Almagor, R.: Internet history. Int. J. Technoethics **2**, 45–64 (2011)
7. Naughton, J.: The evolution of the Internet: from military experiment to general purpose technology. J. Cyber Policy **1**(1), 5–28 (2016)
8. Uskov, V.L.: Mobile software engineering in mobile computing curriculum In: Proceedings of the 2013 3rd Interdisciplinary Engineering Design Education Conference, pp. 93–99 (2013)
9. Tomiyasu, H.: Feature articles: technology for innovating software production software production technologies that support large-scale system development
10. Wang, J., Ren, D.: Research on software testing technology under the background of big data. In: Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 2679–2682 (2018)
11. Sneha, K., Malle, G.M.: Research on software testing techniques and software automation testing tools. In: Proceedings of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), pp. 77–81 (2017)
12. Rahman, A., Sunny, F.H., Mishu, H.M., Sumi, F.: Open access software testing algorithm units. **1**, 271–275 (2017)
13. Sawant, A., Bari, P., Chawan, P.: Software testing techniques and strategies. Int. J. Eng. Res. Appl. **2**, 980–986 (2012)
14. Kumar, D., Mishra, K.: The impacts of test automation on software's cost, quality and time to market. Procedia Comput. Sci. **79**, 8–15 (2016)
15. Srivastava, D.P., Kim, T.-H.: Application of genetic algorithm in software testing. Int. J. Softw. Eng. Appl. **3**, 87–96 (2009)
16. Karnavel, K., Santhoshkumar, J.: Automated software testing for application maintenance by using bee colony optimization algorithms (BCO). In: Proceedings of the 2013 International Conference on Information Communication and Embedded Systems (ICICES), pp. 327–330 (2013)
17. Kaur, M., Kumari, R.: Comparative study of automated testing tools: TestComplete and QuickTest Pro. Int. J. Comput. Appl. **24**(1), 1–7 (2011)
18. Miller, E.: Advanced methods in automated software test. In: Proceedings of the conference on software maintenance 1990, p. 111 (1990)
19. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (1994)
20. Florence, C.M.L.: Deep neural network based hybrid approach for software defect prediction using software metrics. Clust. Comput. **22**(s4), 9847–9863 (2019)
21. Malhotra, R.: Comparative analysis of statistical and machine learning methods for predicting faulty modules. Appl. Soft Comput. J. **21**, 286–297 (2014)
22. Gondra, I.: Applying machine learning to software fault-proneness prediction. J. Syst. Softw. **81**(2), 186–195 (2008)
23. Yang, B., Li, X: A study on software reliability prediction based on support vector machines (2008)
24. Can, H., Jianchun, X., Ruide, Z., Juelong, L., Qiliang, Y., Liqiang, X.: A new model for software defect prediction using Particle Swarm Optimization and support vector machine, In: Proceedings of the 2013 25th Chinese Control and Decision Conference (CCDC), pp. 4106–4110 (2013)
25. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Trans. Syst. Man Cybern Part C **40**(2), 121–144 (2010)
26. Williams, L.J.: Principal component analysis, pp. 433–459 (2010)
27. Abualigah, L. Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. Neural Comput. Appl. **33**, 2949–2972 (2021)
28. Zoppis, I., Mauri, G., Dondi, R.: Kernel Methods: support vector machines. In: Ranganathan, S., Gribskov, M., Nakai, K.,

Schönbach, C.B. (eds.) Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics, pp. 503–510. Academic Press, Oxford (2019)
29. Sayyad Shirabad, J., Menzies, T.J.: The {PROMISE} Repository of Software Engineering Databases (2005)
30. Arar, Ö.F., Ayan, K.: Software defect prediction using cost-sensitive neural network. Appl. Soft Comput. **33**, 263–277 (2015)
31. Abaei, G., Selamat, A., Fujita, H.: An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction. Knowl. Based Syst. **74**, 28–39 (2015)
32. Vandecruys, O., Martens, D., Baesens, B., Mues, C., De Backer, M., Haesen, R.: Mining software repositories for comprehensible software fault prediction models. J. Syst. Softw. **81**(5), 823–839 (2008)
33. Chang, Y.-W., Hsieh, C.-J., Chang, K.-W., Ringgaard, M., Lin, C.-J.: Training and testing low-degree polynomial data mappings via linear SVM. J. Mach. Learn. Res. **11**(48), 1471–1490 (2010)
34. Kumudha, P., Venkatesan, R.: Cost-sensitive radial basis function neural network classifier for software defect prediction. Sci. World J. **2016**, 2401496 (2016)
35. Hudaib, A., Zaghoul, F.A.L., Widian, J.A.L.: Investigation of software defects prediction based on classifiers (NB, SVM, KNN and decision tree). J. Am. Sci. **9**(12), 381–386 (2013)
36. Wahono, R.S., Herman, N.S., Ahmad, S.: Neural network parameter optimization based on genetic algorithm for software defect prediction. Adv. Sci. Lett. **20**(10–12), 1951–1955 (2014)

Mohd. Mustaqeem has completed B.Sc (Hons) in Computer Science and Applications from Aligarh Muslim University, India in 2015. And also has completed his Master of Computer Science and Applications from the same university in the year 2018. After completing the study, he is serving as an Assistant Professor of Computer Science and Engineering in Institute of Technology and Management, Aligarh. Along with teaching, he has appointed as a supervisor of B.tech students for their technical projects. His research area is data analysis, cloud computing, e-learning, Artificial Intelligence and software engineering.

Mohd. Saqib received his B.Sc. in Computer Application from the Department of Computer Science, A.M.U, Aligarh, India in 2015. And also completed a Master of Computer Application and Science from the same university with distinction in 2018. He was a research assistant in the Centre of Advanced Research in Electrified Transportation (CARET), A.M.U and during this period, he worked on projects related to monitoring and accessing smart grid data via cloud computing. Nowadays, he is pursuing M.Tech in Data Analytics from the IIT (ISM) Dhanbad. He has been involved in automation in smart grid and applied artificial intelligence in seismology. His area of interest is data analysis, artificial intelligence, and applied statistics.