WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Kernel Supervised Sub-Code Prediction Method Using To Analysis Software Subclass and Defect Prediction

***V.RUCKMANI[a], S.PRAKASAM[b]***

V.Ruckmani, Assistant Professor, Voorhees College, Anna Salai, Vellore – 1, Tamil Nadu, India
*S.Prakasam, Associate Professor, SCSVMV University, Enathur, Kanchipuram, Tamil Nadu, India*

A B S T R A C T

Software is testing automated methods to improve the consistency of software design model checking. Predicting software defects is the main focus of the engineering community. Many software developers maintain their own software libraries, which help predict software defects. Prediction algorithms are based on machine learning, data manipulation, perceptual prediction and empirical research. The research community still faces some challenges in how to build and there are many research opportunities. In this work, the Kernel Supervised Sub-Code Prediction (KSSP) The technique has been used to detect previous software failures during the testing process. This technology can help us reduce the cost of software projects by reducing software testing. The identification task can provide some practical guidelines for two software engineering researchers and practitioners to predict future software flaws. A brief overview of some popular KSSP methods is provided along with they are suitable for testing software. The Subclass Component Analysis (SCA) is subcategory indicators are set, it is close to predicting defects between projects. Our conclusion of this work is that it is possible to quickly transfer an understanding of defect prediction lessons, even if the project uses a different set of indicators.

* Corresponding Author Tel: 9994203662
*E-Mail Address :* rukkhue@gmail.com

## 1. Introduction

Defects are errors in the software system that causes the system to behave correctly or produce unexpected results. Repairing defects typically consumes about 80% of the total budget of a software project. This method can significantly reduce costs if the defects are fixed early. Therefore, defect prediction models are typically used for priority quality improvement and defect avoidance efforts. Despite the rapid growth of software development, software has many flaws for a variety of reasons. In the software development process, software testing is the main stage for reducing software defects. If developers and testers can correctly predict software flaws, it can save money, time and energy. It is necessary to perform a comparative analysis of defect prediction of mining software based on the classification rules. It chose different classification algorithms for comparison and developed a scheme for this purpose. The assessment analyzes the predictive performance of the learning scheme for a particular dataset in the past of competition.

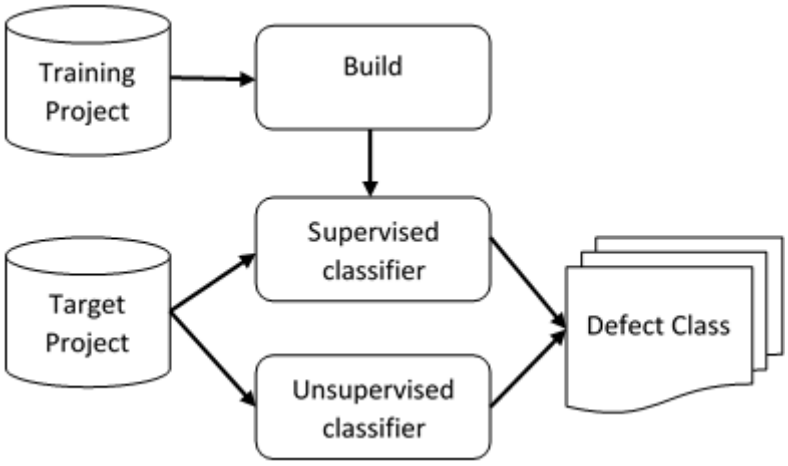


**Fig.1 Illustrated process of project defect analysis**

Code in a large code base contains errors and requires extensive testing and code review issues. On the one hand, code review is an effective way to find errors early in development. On the other hand, with modern code inspection tools, even if it takes the time of an expert programmer, they can use it effectively in other ways. The right amount of code review is balanced with effort. Otherwise, you will miss the wrong price to enter the review code. Identifying problem domains is a way to make code reviews more effective.

Building high quality software with a limited quality assurance budget is becoming more difficult. Various software prediction models are used these days to predict failures from software metrics. Software failure prediction, which is predicted to be a failure-prone module and allows limited resources to be allocated before verification and verification activities or software is released. At an early stage, accurate failure prediction is a better way to reduce testing effort.

Such predictions can be used for test and debug optimization where resources, more resources, should be allocated to multiple defect-prone modules. Predicting project defects is a strategy. Trains belong to other project data generalization prediction models, and the models are used to predict trends in defects that are part of the project of interest. However, in the machine learning literature, many composite technologies have suggested a combination of multiple classification modes.

Reusable component-oriented software infrastructure and integrated frameworks for reuse are the composition of these components, the components of which can provide specific functionality such as word processors and spreadsheets. It is the type of software component used in reuse-oriented software processing is web services, and service standards are used to develop these standards and can be used remotely. The daily use of existing solutions in the development of new systems is an important attribute of the discipline of all mature technologies. Software reuse is a development in various application areas such as telecommunications, factory automation, automotive, aero electronics and practices. Software engineering produces a number of technologies and methods that facilitate the development of complex software systems for software reuse.

## 2. Related work

In this [1], the author investigates the four most relevant software engineering activities, depending on the effectiveness of the tracking requirements. These refer to the requirements-level impact analysis between requirements and source code. It has proposed some positive steps that may reduce the variance in experimental results [2]. First, it requires reporting protocols to improve the reproducibility of our experiments. Secondly, more joint research may help to overcome the problem of blindness, especially blind analysis, should become a daily practice problem compared with the unskilled application of technology.

Based on 42 major studies that meet our selection criteria, they jointly report 600 sets of empirical forecast results. By reverse engineering, common response variables, we randomly to study four model-building factors (classifiers, datasets, input metrics, and groups of researchers) to predict model performance. Effect ANOVA model was built relative contribution [3]. This article [4] is an application of Fuzzy Analytic Hierarchy Process (FAHP), a popular multi-criteria determination technique for assessing the performance of certain classifiers. Rather than relying on the choice of preferred measures rather than this evaluation framework, combine performance measurements with a wider spectrum to evaluate the performance of the classifier. The aggregation method can measure the correlation between the critical ALTER and the correlation with the number of measurements and defects [5]. To build a model that predicts defect levels or numbers, or to apply only sums or achieve similar performance, often to reach the closest best performance than both other survey aggregation methods. When combining the aggregation methods of all surveys.

Improved subcategory discriminant analysis (SDA) methods have been proposed to improve the subcategory discriminant analysis (SDA) used to achieve equilibrium subcategories [6]. Cross-project prediction, semi-supervised transfer component analysis (SSTCA) methods are used to make the distribution of source and target data consistent, and because the SSTCA + ISDA prediction method has been proposed. In the new software network model [7], each node in the network can create coordinate obstacles that are more orderly than the network graph and assign a set of call information that reflects that function. The combination of different packaging methods includes a support vector machine (SVM) and a maximal correlation (MR) [8] filtering method to find significant indicators and an artificial neural network (ANN). The proposed method of injecting a filter score into the packaging selection process can guide the search process to effectively determine significant indicators. Such a cross-project semi-supervised defect prediction (CSDP), in this case[9]. Supervision of some internal projects Semi-finished defect prediction (WSDP) methods have been developed in recent years, but there is still room for improvement in prediction performance.

In the analysis, software module bad label values, where software measurement parameters are thought to affect factors and independent variables, are thought of as the dependent variable [10]. In classification prediction, the false value of the good module is replaced with a negative value, and the true value of the bad module is replaced with a positive

value. This article proposes a framework for developing fuzzy logic-based software prediction models using different sets of software metrics [11]. Its goal is to provide a set of common indicators for software defect prediction. Introduced in this article to provide an easy-to-understand prediction model and generalize software metrics for defect prediction. The results show [12] the ability of fuzzy logic to generate a transparent defect prediction model. It also evaluates software metric choices to find the most obvious set of metrics. This is because it uses the concept of [13] embedded in the stratification embedded in nearest neighbor (STr-NN) to generate the modified balance and training datasets. For this project, directly use the STr-NN method for defect prediction to reduce the difference in distribution between the source and target datasets.

Based on SimASTToken2Vec, we will design a new unsupervised integration method to learn the representation of the meaning of these extracted token vectors. In the future, bidirectional long-term and short-term memory (BiLSTM) [14], neural networks will be used to learn semantic functions from automatically embedding token vectors. This dataset consists of 10 large open source projects written in the Java programming language. These projects come from different fields of application [15] and have been widely used as experimental subjects in previous software defect prediction (SDP) research.

Over the past few years, some cross-project defect prediction (CPDP) methods have been proposed to reduce the difference in data distribution between the source and target projects. Details of the currently proposed CPDP method can be found in recent meta-analyses and systematic literature reviews [16, 17]. The settings for these methods are consistent with the settings that have been best done in recent comparative studies of different CPDP methods.

This method helps identify the need for immediate attention, so that reliable priority flaws in the software can be dealt with first [18], and the module can be improved more quickly. Our goal in this study is to improve the classification accuracy of data mining algorithms. One solution to this problem is the development of [19, 20] through standard guidelines for systematic data collection processes provided to open communities, thereby reducing data collection bias. This article presents a tool for demonstrating systematic data collection programs from defect prediction datasets in open source bug tracking and source code control warehouse software.

## 3. Implementation of the proposed Kernel Supervised Sub-Code Prediction

Machine learning algorithms especially the Kernel Supervised Sub-Code Prediction method is suitable for adapting the data to the behavior of the target program based on (debugging) observations. In this software defect prediction problem, classification provides a subset of known datasets of training classes on which it is executed and unknown classes of datasets are tested against the model. Cross-validation is used to divide a dataset into complementary subsets. It validates the analysis of another subset of the test set, called the analysis of a subset of the running training set.

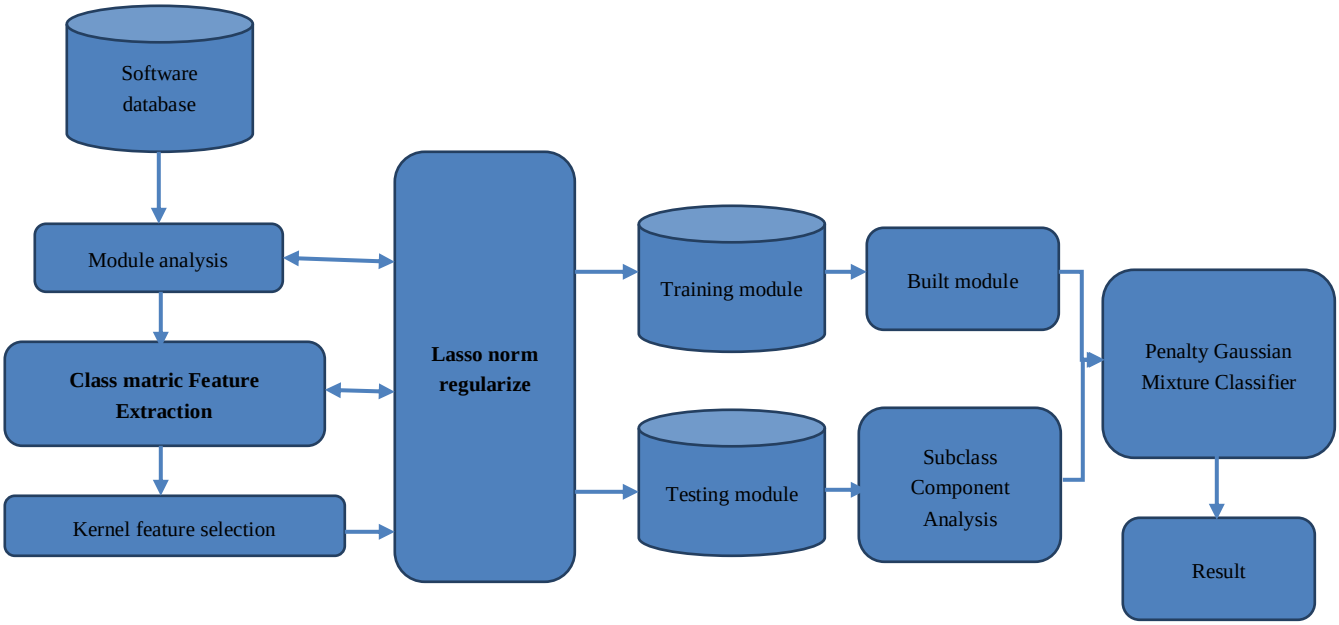


**Fig.2 Proposed Block Diagram of the KSSCP**

A Subclass Component Analysis (SCA) using in Penalty Gaussian Mixture Classifier (PGMC) approach to predict defects across projects even with code class metric sets. In this block diagram of proposed method Kernel Supervised Sub-Code Prediction shown in figure 2. It uses probabilities to express explicit modeling uncertainties that analyze and track multiple categories of system state. The Penalty Gaussian Mixture Classifier (PGMC) is an effective supervised learning classification algorithm used to classify N-dimensional datasets. The pattern in our SCA is marked by routine deficiencies in the transfer of knowledge classification. It

categorizes defects such as source and target data that are set between the target data set by the source, distribution and match code coming from the data set. The training data set, but used as the input of the class label, can be obtained in conjunction with a well-trained model. Finally, the test package is presented to the trained sample and compared with the output of the results.

### *3.1 Feature selection using LASSO Norm Regularize (LNR)*

Overcomes the problem of high dimensional feature space, commonly used for feature extraction and feature selection. It improves classification accuracy by removing unnecessary and inappropriate features. The LASSO algorithm used to select the software feature extraction and selection. LASSO regression has the same multiple linear regression model. The ability of this LASSO is to handle any number of attributes without decreasing its parameter estimate to zero. The LASSO algorithm automatically chooses the relevant features and rejects the others. The optimal feature selection like Line count, Branch count, Derived Halstead in software dataset.

**Algorithm steps:**

Step 1: The input data consider $x_1, x_2, x_3, x_4 \ldots\ldots x_n$, where X is the independent variable, y consider optimal features.

$$y = f(x, k) \qquad \text{--- (1)}$$

Step 2: To estimate the output constant $k$, a series of $n$ measurements with different outputs will produce a set of data. $(x_i, y_i)$ Where i = 1, 2, 3, ….n.

Step 3: In this input data to extract relevant feature (S)

$$s = \sum_{i=1\ to\ n}(y_i - k.x_i) \qquad \text{--- (2)}$$

Step 4: If there is an error in the selection process denote þ this error to estimate.

$$y_i = k.x_i + þ_i \qquad \text{--- (3)}$$

Step 5: The least optimal feature estimate of the output constant, $k$, is given by

$$k = \sum_{i=0}^{x,y} x_i.y_i(s_i)/\sum_i x_i^2 \qquad \text{--- (4)}$$

The above generated least optimal features estimate is the combination obtained equation 2 and 3.

Step 6: The output constant k is the same value required by LASSO to get the same value as in equation (4).

To represent this simple scheme, all candidate feature sets are binary representations that are considered binary genes. Each feature also contains a fixed-length string in binary format that represents a particular subset of a given feature set. The performance of all selected feature subsets is calculated using the merit function and the classification results are also calculated. The optimal subset of features were given as the output of the best set of classification features that could be used to train a prediction system.

### *3.2. Classifier using Subclass Component Analysis*

In this Subclass Component Analysis used Penalty Gaussian Mixture Classifier (PGMC) for classify the predictive subclasses in software defect analysis. The Penalty Gaussian Mixture Classifier is a supervised learning algorithm and mathematical model that can analyze data and recognize patterns. The classification process is divided into two stages, one is the training stage and the other is the testing stage. When the data is categorized during training, the labels assigned to the data during the test phase are provided as input. The accuracy of the classification depends on the efficiency of the training model.

For a set of training samples, the ML estimation finds the model parameters which maximize the likelihood of the PGMC. Given a sequence of N training vectors $x = \{\overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}, \overrightarrow{x_n},\}$the PGMC likelihood is written as in Equation 5

$$P\left(\frac{x}{\lambda}\right) = \pi_{i=0}^{n} P\left(\overrightarrow{x_i}|\lambda\right) \qquad \text{----- (5)}$$

As this is a nonlinear function of λ, direct maximization is not possible. ML parameters can be estimated iteratively using the expectation – maximization algorithm. The EM algorithm begins with an initial model λ and estimates a new model $\lambda, such\ that\ P\left(\frac{x}{\lambda}\right) \geq P(x|\lambda)$. The new model becomes the initial model for the next iteration.

Mixture weight = $\overrightarrow{p_i} = \frac{1}{n}\sum_{i=0}^{n} p(j|\overrightarrow{x_i}, \lambda)$ ---- (6)

**Algorithm steps**

**Input:** dataset predict variable, density *g*, velocity *v*, defect finding time t.

**Output:** Defect classification result

Feature software datasets ($M_{1\text{-}n}$)

Determine (*g*, *v*, *t*);

**For** *each i to n* do

**For** *each i to n-1* do

**If** *M* (*g*, *v*, *t*) not terminate **then**

**For** every ($M_{1\text{-}n}$), recheck $\int_{i,j}^{n}(n(i), n(j))$to follow (*g*, *v*, *t*) **do**

**If** *M* (*g*, *v*, *t*) determined **then**

Build a model = *f* (*g*, *v*, *t*)

**End if**

**End for**

**End if**

Data set divide to two parts there are train (80%) and test (20%) *M (I,j)*

Train data→Tdata and Test data→Vdata

Train model with *M* (*i*; *j*) (80%)

Validate model with *M* (*i*; *j*) (20%)

**If** *M* (*i*; *j*) equal D successful **then**

Determine prediction maximum likelihood parameters using equation 5.

To apply the weight using a equation 6 and apply the Penalty optimization function to predict the defect classes.

**Else**

Clean and pre-process dataset ($M_{1\text{-}n}$)

Again

**End if**

**End for**

**End for**

**End procedure**

This is a useful supervised learning classification algorithm that can be used to classify N-dimensional datasets. During the training phase, PGMC is designed for each type of data. In this PGMC must be configured to contain defective and non-defective data samples. There must be no interaction between different predictive subcategories PGMC. At the classification stage, unknown levels of data are given by the PGMC as an input penalty function for each class to optimize the classifier. The predicted class is associated with the maximum probability of PGMC and subclass component analysis.

## 4. Result and discussion

Statistics provide a strong basic background for quantifying and assessing results. This section presents the results of a work that proposed a technique for predicting software defect analysis using machine learning. The table 1 show the simulation parameters of proposed method to used analysis the performance. In this proposed method, evaluate the following equations 7, 8, 9, 10: Classification accuracy, Precision, Recall, F1 score and time complexity.

**Table 1 Simulation Parameters**

| Parameters | Value |
|---|---|
| Tool | Visual studio |
| Dataset name | NASA MSP repository |
| Number of data | 1000 |

- $CA = \frac{Number\ \ of\ classified\ \ files}{number\ \ of\ files} * 100$ ---- (7)
- $Precision = \frac{true\ positive}{true\ positive + false\ positive}$ ---- (8)
- $Recall = \frac{true\ positive}{true\ positive + false\ negative}$ ---- (9)
- $F1 = \frac{precision * Recall}{precision + Recall}$ ---- (10)

The NASA dataset used to input data to our system it's a suitable for high data support and dataset are labeled and unlabeled classes. In this analysis of proposed Kernel Supervised Sub-Code Prediction (KSSP) method compare to existing method Support Conventional neural network (CNN), Defect prediction via attention-based recurrent neural network (DP-ARNN), Random Forest (RF) and Improved Subclass Discriminant Analysis(ISDA)method to analysis the performance.

**Table 2 Analysis of Proposed Method Prediction Performance**

| Methods | Classification Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| Random forest | 0.669 | 0.75 | 0.79 | 0.694 |
| CNN | 0.712 | 0.80 | 0.82 | 0.746 |
| ISDA | 0.87 | 0.83 | 0.85 | 0.73 |
| KSSP | 0.90 | 0.89 | 0.87 | 0.86 |

Table 2 shows a comparison of the existing and proposed method disease prediction performance. In this result, table 2 shows the classification accuracy, precision, recall, f1 score are analysis proposed method KSSP and existing methods SVM, Random forest, Bayesian classifier.

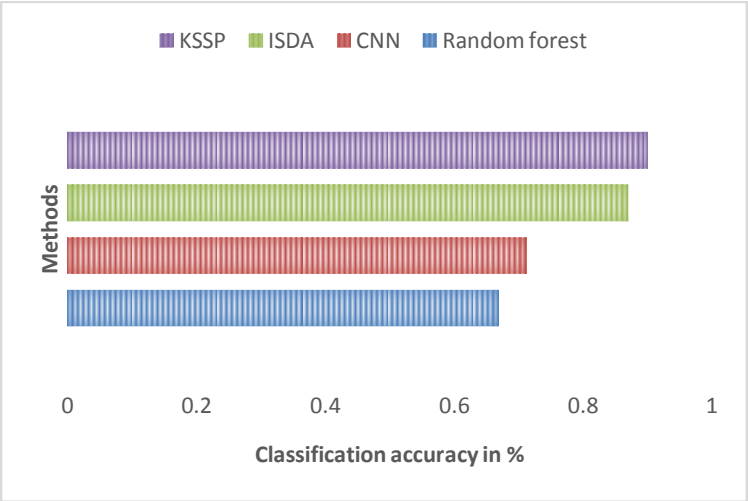


**Fig.3 Analysis's Classification Accuracy**

Our machine learning method evaluation result provides a higher performance of classification accuracy compared to other existing method its show in figure 3. The proposed KSSP method have a 90% classification and existing method Random forest has 66%, CNN have 71%, and ISDA have 87% classification accuracy.

Time calculation parameter is an input of the number of predicted class based on the algorithm request parameter calculated using the database is listed, as follows is represented the execution time of each algorithm.

QA= query analytics

T= trust

A= accuracy

DA= Number of database function

N=no of non-class function

Time $= (QA + T + A)/ (DA + N)$

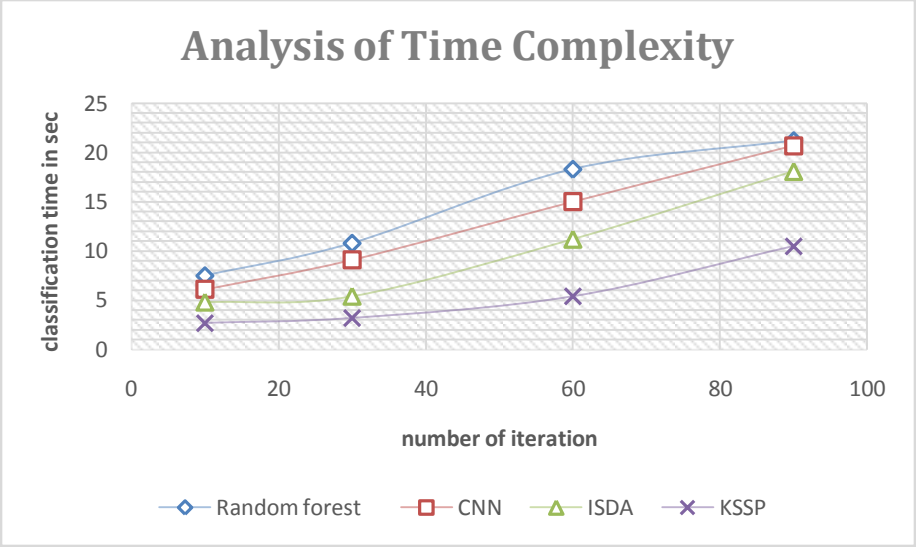


**Fig.4 Analysis of Time Complexity**

In this result of time complexity taken number prediction iteration, the proposed method KSSP is 10.5sec classification time for 90 iteration; similarly, the existing method RF, CNN and ISDA provide an 21.2sec, 20.7sec, and 18.1sec of classification time for 90 iteration it comparison is shown in figure 5.

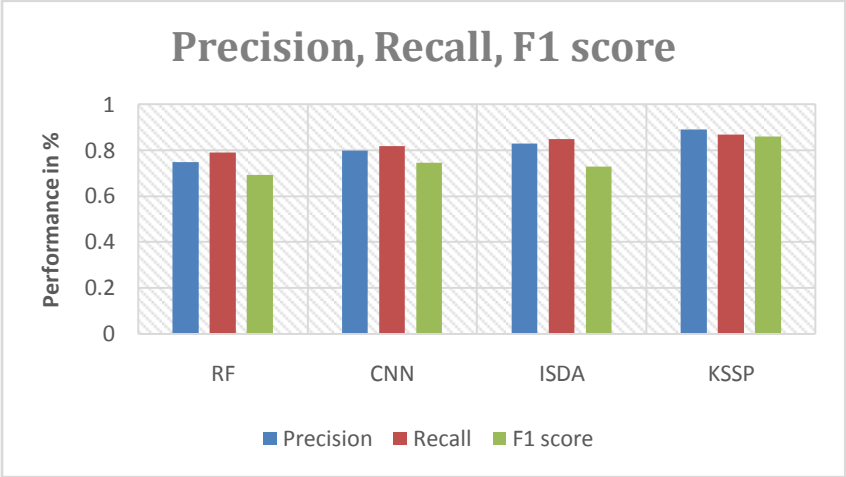


**Fig.5 Analysis of Precision, Recall, and F1 score**

The proposed KSSP method provides a 0.86% of F1 Score, 0.89% of precision, and 0.87% of recall values. Similarly the existing method RF, CNN, and ISDA have 0.75%, 0.8%, and 0.83% of precision rate compare to proposed method. In this analysis of precision, recall, and f1 score of existing method and proposed method comparison is shown in figure 5.

## 5. Conclusion

The ability to identify software defects early in development is important to reduce costs and improve the overall efficiency of the testing process. Most software system failures were found in some of their own components. To improve the efficiency and quality of software development, you can use the benefits of data mining and analysis to predict the large number of software developments in your collected defect data. In this proposed Kernel Supervised Sub-Code Prediction (KSSP) method to analysis the software kernel class function. In this proposed method provide a more efficiency and defect prediction accuracy compare to others.

## Reference

[1]. Rempel, P., & Mader, P. (2017). Preventing Defects: The Impact of Requirements Traceability Completeness on Software Quality. IEEE Transactions on Software Engineering, 43(8), 777–797.

[2]. Shepperd, M., Hall, T., & Bowes, D. (2017). Authors' Reply to "Comments on 'Researcher Bias: The Use of Machine Learning in Software Defect Prediction'." IEEE Transactions on Software Engineering, 1–1.

[3]. Shepperd, M., Bowes, D., & Hall, T. (2014). Researcher Bias: The Use of Machine Learning in Software Defect Prediction. IEEE Transactions on Software Engineering, 40(6), 603–616.

[4]. Moh'd, Hussam & Dichter, Julius. (2019). Applying the FAHP to Improve the Performance Evaluation Reliability of Software Defect Classifiers. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2915964.

[5]. Zhang, F., Hassan, A. E., McIntosh, S., & Zou, Y. (2017). The Use of Summation to Aggregate Software Metrics Hinders the Performance of Defect Prediction Models. IEEE Transactions on Software Engineering, 43(5), 476–491.

[6]. Jing, X.-Y., Wu, F., Dong, X., & Xu, B. (2017). An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems. IEEE Transactions on Software Engineering, 43(4), 321–339.

[7]. Ai, J., Su, W., Zhang, S., & Yang, Y. (2019). A Software Network Model for Software Structure and Faults Distribution Analysis. IEEE Transactions on Reliability, 1–15.

[8]. Huda, S., Alyahya, S., Mohsin Ali, M., Ahmad, S., Abawajy, J., Al-Dossari, H., & Yearwood, J. (2018). A Framework for Software Defect Prediction and Metric Selection. IEEE Access, 6, 2844–2858.

[9]. Wu, F., Jing, X.-Y., Sun, Y., Sun, J., Huang, L., Cui, F., & Sun, Y. (2018). Cross-Project and Within-Project Semisupervised Software Defect Prediction: A Unified Approach. IEEE Transactions on Reliability, 67(2), 581–597.

[10]. REN, J., & LIU, F. (2019). Predicting Software Defects Using Self-Organizing Data Mining. IEEE Access, 1–1.

[11]. Al-Jamimi, H. A. (2016). Toward comprehensible software defect prediction models using fuzzy logic. 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS). doi:10.1109/icsess.2016.7883031

[12]. Gong, L., Jiang, S., Bo, L., Jiang, L., & Qian, J. (2019). A Novel Class-Imbalance Learning Approach for Both Within-Project and Cross-Project Defect Prediction. IEEE Transactions on Reliability, 1–15.

[13]. H. Wan, G. Wu, C. Ming, H. Qing, W. Rui, and Y. Mengting, "Software defect prediction using dictionary learning," in Proc. Int. Conf. Softw. Eng. Knowl. Eng., Pittsburgh, PA, USA, Jul. 2017, pp. 335–340.

[14]. Chen, D., Chen, X., Li, H., Xie, J., & Mu, Y. (2019). DeepCPDP: Deep Learning Based Cross-Project Defect Prediction. IEEE Access, 7, 184832–184848.

[15]. X. Chen, D. Zhang, Z.-Q. Cui, Q. Gu, and X.-L. Ju, ''Dp-share: Privacypreserving software defect prediction model sharing through differential privacy,'' J. Comput. Sci. Technol., vol. 34, no. 5, pp. 1020–1038, 2019.

[16]. S. Hosseini, B. Turhan, and D. Gunarathna, ''A systematic literature review and meta-analysis on cross project defect prediction,'' IEEE Trans. Softw. Eng., vol. 45, no. 2, pp. 111–147, Feb. 2019.

[17]. S. Herbold, A. Trautsch, and J. Grabowski, ''A comparative study to benchmark cross-project defect prediction approaches,'' IEEE Trans. Softw. Eng., vol. 44, no. 9, pp. 811–833, Sep. 2018.

[18]. K.PUNITHA, Dr. S. CHITRA, Software Defect Prediction Using Software Metrics - A survey, IEEE 2013, pg.no: 555 – 558.

[19]. Goran Mauˇsa, Tihana Galinac Grbac, Bojana Dalbelo, Software Defect Prediction with Bug-Code Analyzer - a Data Collection Tool Demo, IEEE 2015, pg.no: 1-2.

[20]. Ke Liang, Juan Guo and Jinghua Wang, On-Board Software Maintenance for Manned Spacecraft, IEEE 2014, pg.no: 235 – 239.