International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-5, June 2019

Prediction of Cost and Defects in Software Development using Bayesian and Subspace Algorithms

Sita Kumari.Kotha, Suhasini. Sodagudi, Anuradha . T

Abstract: ***Software Development is the discipline of initiating, organizing, executing, controlling and completing the project work of a group to accomplish target and meet progress. Prediction of software defects plays an important role while building high quality software. Machine learning algorithms are utilized in software development for better Performance. Machine learning algorithms have proven to be of great practical value in a variety of application domains. They are particularly useful for poorly understood problem domains where little learning exists to develop powerful algorithms and for the domains where there are expensive databases containing valuable implicit regularities to be discovered. Machine learning is a kind of Artificial Intelligence (AI) that enables programming applications to end up more exact in expectation results. The main objective of this paper is to predict the cost and defects of a project or an application in an efficient manner by applying machine learning algorithms. The Bayesian and subspace algorithms are implemented to predict the defects and to make decisions whether the project can be continued or not. Two algorithms are compared and the results are exhibited by applying on software defect data set. .***

Index Terms: ***Software Development, Machine Learning, Bayesian Algorithm, Subspace Algorithm.***

I. INTRODUCTION

Enhancing defect free software is strenuous as there will be untold bugs where the fundamental concepts of the software development methodologies need to be employed. Because of some software modules, the deployment stage of projects will be burden for the users and high for enterprises. So, estimating the defective modules or files in a software is a definitely a challenging task, since it assists in minimizing the total cost of the software and increase in success rate. Estimation of deficiency provides many choices to the development team by re-examining for which the defectiveness estimation is high. By examining defective modules, the resources of the project were employed better which results the deployment of project very easy for both users and project team.

Manuscript published on 30 June 2019.
***** Correspondence Author (s)
Sita Kumari.Kotha, Dept. of Information Technology, Velagapudi Ramakrishna Siddhartha Engineering College, Vijyawada, Kanuru, India
Suhasini. Sodagudi, Dept. of Information Technology, Velagapudi Ramakrishna Siddhartha Engineering College, Vijyawada, Kanuru, India
Anuradha . T, Dept. of Information Technology, Velagapudi Ramakrishna Siddhartha Engineering College, Vijyawada, Kanuru, India

Even there are many literature studies, defect estimation is quite difficult to solve. There are many conclusions about description of defects which leads to wrong issues. This can be understood in a better way when defects that were described were recognized [1]. The vital task in Software Project Management is estimation of software development effort. Many designs were suggested to build a correlation in between software size and effort. But still many issues were present due to project data, available of the project in the initial phases is usually half-done, inconsistent, pending and unclear [2]. The effort estimates are applied as input to project plans, iteration plans, budgets, investment analyses, cost processes to get precise evaluations. The main aim of this paper is to estimate the required effort to build a software artifact. Over grazing of this effort may cause the tight plan of software development and issues may be left in the system after delivery, whereas under estimation might cause a delay in system delivery and may get complaints from customers [3]. The significance of this process has stimulated the development of prediction models to evaluate the required effort as exact as could be expected under the circumstances. The basic idea of applying machine learning strategies for effort prediction is that the historical dataset contains numerous historical projects which are portrayed by features with their values. These projects and the values can be utilized for similar projects in estimating the project efforts. The aim of Machine Learning techniques is to grasp the deep-rooted pattern of feature values and their relations with project efforts, which can be utilized for evaluating the effort of new projects [4].

II. LITERATURE REVIEW

In the year 1996 the authors M. Shepperd and C. Schofield followed the technique to evaluate by analogy to report the project for which the evaluation was done and to use this version to detect similar projects which were completed. The cost values for these projects were used to build and evaluate the new project. Analogies are detected by evaluating Euclidean distance in n-dimensional space where each measurement relates to a variable.

The values should be standardized because each measurement must contribute parallel weight to the revelation of discovering analogies [5]. The information required cannot be categorical and it works only with binary values. This may be due to a lesser degree a confinement than it initially shows up since such information might be better utilized to split huge datasets to homogeneous ones.

The project for which the cost is to be estimated is compared with the similar projects that were executed previously to check whether this type of project was done previously or not. If such project exists, required data is taken to develop the project. If multiple projects are available, then a project will be chosen based on the user requirements to estimate the effort. The benefit of this approach is that assessments can be made based on actual project, estimator's past involvement and data. The impact of the project can be estimated by comparing the existing project with the one to be developed. The main drawback of this approach is one should know how much close it is and whether it is certain to put in the analogies as very few might lead to actual projects being used.

A. Machine Learning Algorithms for Software Project Time Prediction

In the year 2015 the authors W Han, H Jiang, X Zhang, W Li determined that the Software Project Management (SPM) is one of the important feature to accomplish whether code can be executed successfully or failures are existing. Estimation of code improvement time is the key assignment for the effective SPM. The precision and reliability of estimation procedures are also equally important. Many Machine Learning methods were compared to predict the time required to develop the software [6]. To perform this, the measures like MRE, MMRE and MDMRE are used. Algorithms ranking was made based on the values of the metrics of interest. Table 1 depicts the values of the algorithms LR, MLP, M5P, GP, LMS and RT.

Table 1 : Comparison of Algorithms

	LR	MLP	M5P	SMO	GP	LMS	RT
MRE	0.91	0.921	0.897	0.956	0.976	0.898	0.921
MMRE	0.875	0.936	0.934	0.895	0.951	0.921	0.937
MdMRE	0.915	0.899	0.911	0.901	0.96	0.932	0.945

The algorithms are processed on the project data to compute Correlation Coefficient with which software model for defect prediction can be built and compared. It is observed that Gaussian Process (GP) is giving better result when compared to other algorithms. The main disadvantage is that GP has to consider the entire samples for prediction and may be less effective when the number of features increases.

B. Machine Learning algorithms for Project Management Performance Prediction

In the year 2017, author Sefa O.saglam said that organizations firmly depend on decision support systems and the working hours has been extended as the significance and pro-activeness of the project manager is increased. Agile Project management strategy focus on priorities related to problems and approaches. To solve difficult problems incremental iterative or sprints can be developed. What makes agile an important matter of risk assessment is also carried out to minimize the effect of anticipated risks in the given activity and provides a steady process [7].

III. Related Work

Prediction of defects in software development plays an important role in developing the software. The quality of the software can be improved by identifying the defects as early as possible. At present scenario, software modelling equips gateway to different circumstances like system adjusting to situations, areas of study and no existing processes available for developing powerful algorithmic methods [8, 9]. A procedure is transformational programming, in which the software is built, redesigned and supported at a certain level and then instinctively modified in production quality software through automatic program synthesis [10]. This software development model will allow software engineering to become the discipline to acquire and automate the currently undocumented domain and design knowledge [11]. Software developers distribute knowledge-based application generators instead of non-modifiable application programs.

Machine learning manages the topic of how to assemble computer programs that enhance their performance at some function through experience [12]. Machine Learning principally focuses on the target data. The study involves understanding the target data from training sample data which includes numerous topics. Machine leaning procedures were utilized in enhancing better tools or software products.

A. Software defect prediction

Software Defect Prediction (SDP) is one of the most helpful activities, which identifies the modules with defects. The testing methods can be applied in an organized manner by recognizing the defected modules. The output of the Defect prediction produces the catalogue of defect-prone areas so that quality team allots services to minimize the defects [13]. An increase in the size of the projects makes the defect estimation methods to play a significant role in encouraging programmers to develop more reliable software applications. Software defect estimation is an extremely valuable to check the quality and maintenance effort before software systems were deployed [14, 15].

Bayesian networks provide are extremely reasonable and useful procedure for software defect prediction. It is recommended to build a Bayesian network that reflects all software development activities like analysis, design, coding, testing and maintenance. Consider Bayesian network generation in three steps:

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

(1)Define Bayesian network variables; characterize the underlying connections among the network variables.
(2) Generate the probability distribution of each variable in the network.
(3) Calculate the joint probability distribution of the hypothesis variable.

B. Project effort (cost) prediction

Prediction of cost for a software application is a primary concern in software development. The accurate estimation is hazardous for both programmers and end users because underestimation may result in poor quality of the software and overestimation may increase the budget which results in allocation of many sources which were not necessary [16]. A software project generally doesn't get fail in implementation but they fail if the planning and estimation is not done correctly. A viable approach to handle effort estimation is Bayesian Belief Networks (BBN). BBN contributes its best for circumstances where an algorithmic model is not applicable for prediction. BN(Bayesian Network) are utilized for estimation by demonstrating the flow of the development procedures and by their communications.

IV. Experimental Work

A. Proposed Architecture

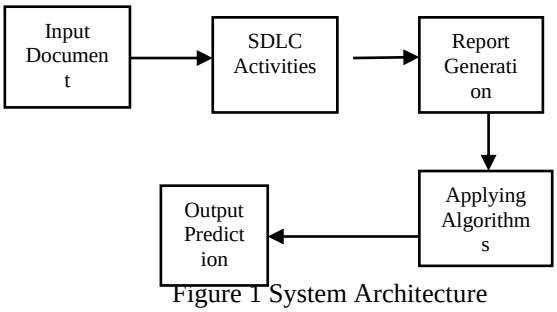

Figure 1 System Architecture

Figure 1 provides the overall proposed framework describing the process.

Steps:

- The benchmark dataset (Project data) is taken into consideration
- The dataset considered here is a software development dataset and SDLC is an agile model.
- The dataset is molded in the form of input and synthesized into agile model phases and machine algorithms are applied which results the prediction of defects in software module.

Machine Learning Algorithms

A. The Bayesian Algorithm:

The BN defect and effort estimates are allowed to hold the right approach. This represents the combined probability of a set of variables. This is done by specifying
(A) Direct Acyclic Graph (DAG) where there are variables in Arrowheads and Arcs correspond to Conditional Independence (sample data area of the study area) and
(B) A group like a local conditional potential table [12, 17].

Considering the observed value of other variables, the BN can be used to summarize the probability distribution for a target variable. Traditionally, BN is applied to any group of variables to calculate probability distribution, in which the value or distributor is given for any group of remaining variables.

The Bayesian Algorithm Steps:

1. Identify the variables of BN that can be in the form of:
i) Theoretical variables to calculate the probability distributions
ii) Data variables
iii) Mediation variables
2. Describe the appropriate random relationships between the variables, in which the relationships should capture and reflect the causality present in the SDLC processes. They were shown as arcs in the corresponding DAG.
3. Calculate the probability distribution for each variable. The complete joint probability distribution for all defective variables is represented in the DAG structure and in a group of conditional probability tables.

B. Subspace Algorithm

The subspace algorithm is implemented to overcome the disadvantages of Bayesian algorithm; it uses K-local hyper plane distance nearest neighbor (HKNN) algorithm to obtain results of defect and cost predictions. It was theoretically shown that for large number of samples in the training set, the algorithm exhibits good performance. In particular, the error committed by the subspace algorithm is at most twice the Bayesian error.

Subspace Algorithm Steps:

- The Dataset was sent as input and a network (Direct Acyclic Graph) is formed to take the data from the dataset and to identify the defects in the dataset.
- The Defects were identified in such a way that if the 'update' attribute was present in analysis or design phases in phase-1 then they were identified as the minor defects and if arises in phase-3 that is during analysis or design or coding or testing, it is identified as major defects.
- The increase in defects will be directly proportional to the increase in cost.
- Initially the minor defects were calculated and applied directly to the cost factor and later the major defects were calculated.
- Depending upon the defects the cost was updated, and if the provided budget is lesser than the calculated cost then it is better to stop developing its next version and reduce defects in the current version.

V. Results and Analysis

The project data set comprising of no of people involved, no of analysers, designers, developers and testers is taken for the existing system for which the machine learning algorithms are applied.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

Table 2 Report Generation

Software Phase Data

Analyzer Initiated @ 2018/02/28 15:08:53

Software Build Cycle Between2013-10-19 to 2014-10-19

WorkLoad Scheduling Across SDLC Teams	
Analyzer's Size	22
Designer's Size	15
Developers's Size	188
Tester's Size	78
Deployer's Size	9
Total's Size	312
Work Days Alloted	366
Work Days Scheduled	358

Information regarding day to day assessment of the developing project is also provided and the defect and cost predictions are calculated by implementing machine learning algorithms

Table 3 Prediction of Defect and Cost applying Bayesian Algorithm

SDLC Stage	Minor Defects	Major Defects	Cost
Analysis	1716	2814	52
Design	3685	2706	36
Coding	12939	7080	195
Testing	4550	6942	114
Maintenance	84	132	0

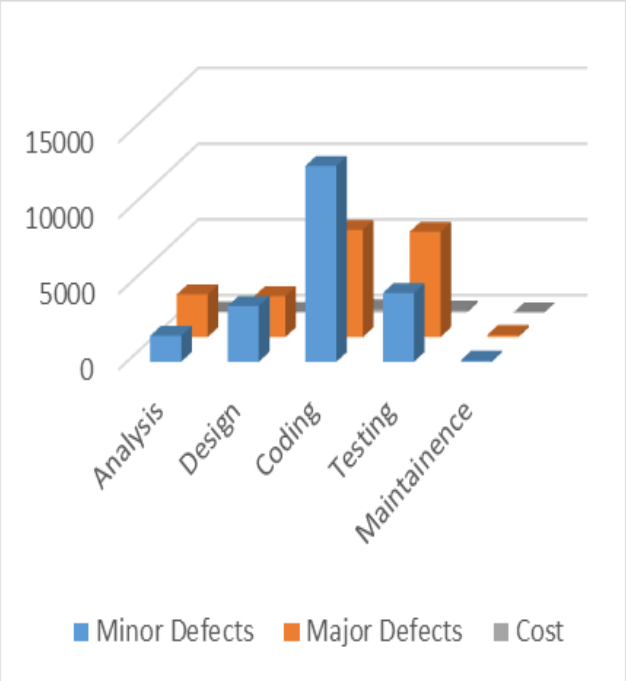

Fig 2 Bayesian Algorithm applied on 1 year Dataset

Table 4. Prediction of Defect and Cost applying Bayesian Algorithm

SDLC Stage	New (Minor)	Update (Minor)	New (Major)	Update (Major)	Cost
Analysis	6076	1522	3660	2460	10
Design	4425	1353	3024	2436	10
Coding	27116	2315	13620	6474	10
Testing	23230	2723	13800	6306	10
Maintenance	1	105	1	138	0

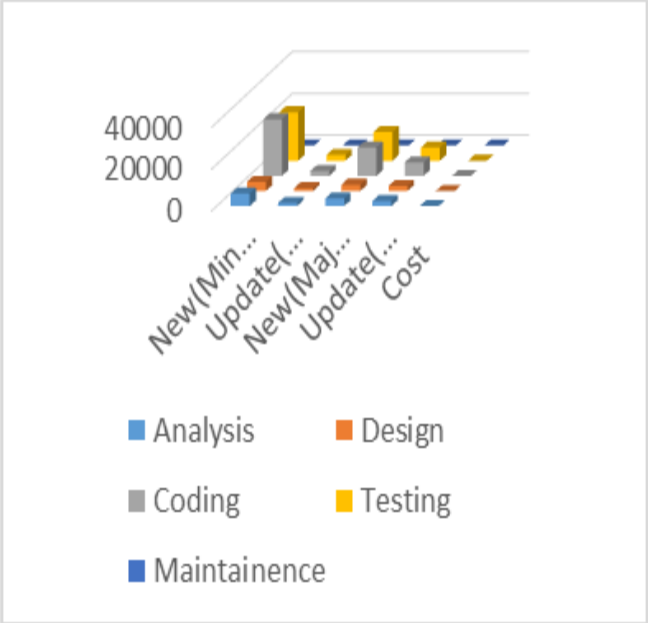

Fig 3 Subspace Algorithm applied on 1 year Dataset

A. Sample Dataset:

```
2013-10-19,New ScenarioSpecs(0),ModuleID:1,Guy.Long,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:2,Dwight.Sutton,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:3,Alex.Warren,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:4,Nichole.Parks,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:5,Edgar.Jimenez,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:6,Alex.Warren,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:7,Alex.Warren,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:8,Edgar.Jimenez,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:9,Joshua.Byrd,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:10,Clay.Salazar,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:11,Chelsea.Gross,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:12,Clay.Salazar,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:13,Edgar.Jimenez,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:14,Neil.Copeland,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:15,Dwight.Sutton,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:16,Alejandro.Cook,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:17,Alex.Warren,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:18,Neil.Copeland,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:19,Chelsea.Gross,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:20,Edgar.Jimenez,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:21,Alejandro.Cook,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:22,Alex.Warren,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:23,Alex.Warren,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:24,Guy.Long,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:25,Eula.Mcdonald,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:26,Dwight.Sutton,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:27,Alex.Warren,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:28,Dwight.Sutton,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:29,Guy.Long,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:30,Theodore.Garza,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:31,Bryant.Robinson,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:32,Bryant.Robinson,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:33,Theodore.Garza,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:34,Alejandro.Barnes,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:35,Nichole.Parks,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:36,Eula.Mcdonald,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:37,Joshua.Byrd,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:38,Alejandro.Barnes,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:39,Clay.Salazar,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:40,Joshua.Byrd,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:41,Edgar.Jimenez,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:42,Clay.Salazar,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:43,Chelsea.Gross,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:44,Edgar.Jimenez,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:45,Dwight.Sutton,Analysis,Phase1
2013-10-19,New CostSpecs(0),ModuleID:46,Bryant.Robinson,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:47,Jesus.Garcia,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:48,Edgar.Jimenez,Analysis,Phase1
2013-10-19,New RequirementsSpecs(0),ModuleID:49,Theodore.Garza,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:50,Alex.Warren,Analysis,Phase1
2013-10-19,New ScenarioSpecs(0),ModuleID:51,Alejandro.Cook,Analysis,Phase1
```

Table 6 Sample Dataset 2

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

```
2013-10-31,Update ScenarioSpecs(4),ModuleID:180,Neil.Copeland,Analysis,Phase1
2013-10-31,Update CostSpecs(4),ModuleID:181,Alejandro.Barnes,Analysis,Phase1
2013-11-01,Update ScenarioSpecs(4),ModuleID:182,Joshua.Byrd,Analysis,Phase1
2013-11-01,Update ChenER's(4),ModuleID:17,Monique.Leonard,Design,Phase1
2013-11-01,Update UML's(4),ModuleID:18,Lillian.Watson,Design,Phase1
2013-11-01,New Presentation(UI)-Logic,ModuleID:1 From (New ScenarioSpecs(1)),Noel.Hudson,Coding,Phase2
2013-11-01,New AutomatedTesting(UFT),ModuleID:1 From (New ScenarioSpecs(1)),Hattie.Simmons,Testing,Phase2
2013-11-01,New Business-Logic,ModuleID:2 From (New ScenarioSpecs(0)),Tony.Mccarthy,Coding,Phase2
2013-11-02,New ManualTesting,ModuleID:2 From (New ScenarioSpecs(0)),Chris.Drake,Testing,Phase2
2013-11-02,Update ChenER's,ModuleID:19,Sheldon.Grant,Design,Phase2
2013-11-02,New Business-Logic,ModuleID:3 From (New ScenarioSpecs(0)),Andy.Johnston,Coding,Phase2
2013-11-02,New AutomatedTesting(UFT),ModuleID:3 From (New ScenarioSpecs(0)),Santos.James,Testing,Phase2
2013-11-02,New Business-Logic,ModuleID:4 From (New ScenarioSpecs(0)),Terrell.Sharp,Coding,Phase2
2013-11-03,New ManualTesting,ModuleID:4 From (New ScenarioSpecs(0)),Guillermo.Davidson,Testing,Phase2
2013-11-03,New Business-Logic,ModuleID:5 From (New ScenarioSpecs(0)),Erick.Love,Coding,Phase2
2013-11-03,New AutomatedTesting(UFT),ModuleID:5 From (New ScenarioSpecs(0)),Willis.Santos,Testing,Phase2
2013-11-03,Update CostSpecs,ModuleID:183,Alex.Warren,Analysis,Phase2
2013-11-03,New Business-Logic,ModuleID:6 From (New ScenarioSpecs(0)),Peter.Mitchell,Coding,Phase2
2013-11-03,New ManualTesting,ModuleID:6 From (New ScenarioSpecs(0)),Jordan.Bell,Testing,Phase2
2013-11-04,New Presentation(UI)-Logic,ModuleID:7 From (New ScenarioSpecs(0)),Manuel.Wright,Coding,Phase2
2013-11-04,New AutomatedTesting(UFT),ModuleID:7 From (New ScenarioSpecs(0)),Gene.Weber,Testing,Phase2
2013-11-04,New Presentation(UI)-Logic,ModuleID:8 From (New ScenarioSpecs(0)),Eduardo.Fuller,Coding,Phase2
2013-11-04,New AutomatedTesting(UFT),ModuleID:8 From (New ScenarioSpecs(0)),Richard.Ferguson,Testing,Phase2
2013-11-04,Update Flows,ModuleID:20,Alberta.Ferguson,Design,Phase2
2013-11-04,New Presentation(UI)-Logic,ModuleID:9 From (New ScenarioSpecs(4)),Sheryl.Lowe,Coding,Phase2
2013-11-04,New ManualTesting,ModuleID:9 From (New ScenarioSpecs(4)),Timmy.Farmer,Testing,Phase2
2013-11-05,Update RequirementsSpecs,ModuleID:184,Neil.Copeland,Analysis,Phase2
2013-11-05,Update Business-Logic,ModuleID:10 From (New ScenarioSpecs(0)),Jacqueline.Cross,Coding,Phase2
2013-11-05,Update AutomatedTesting(UFT),ModuleID:10 From (New ScenarioSpecs(0)),Damon.Cortez,Testing,Phase2
2013-11-05,Update Business-Logic,ModuleID:11 From (New ScenarioSpecs(0)),Timothy.Medina,Coding,Phase2
2013-11-05,New ManualTesting,ModuleID:11 From (New ScenarioSpecs(0)),Winston.Fox,Testing,Phase2
2013-11-05,Update UML's,ModuleID:21,Alberta.Ferguson,Design,Phase2
2013-11-06,Update Presentation(UI)-Logic,ModuleID:12 From (New ScenarioSpecs(0)),Jonathan.Montgomery,Coding,Phase2
2013-11-07,Update AutomatedTesting(UFT),ModuleID:12 From (New ScenarioSpecs(0)),Ernest.Boone,Testing,Phase2
2013-11-08,Update Presentation(UI)-Logic,ModuleID:13 From (New ScenarioSpecs(0)),Jonathan.Powell,Coding,Phase2
2013-11-19,New AutomatedTesting(UFT),ModuleID:13 From (New ScenarioSpecs(0)),Hattie.Simmons,Testing,Phase2
2013-11-10,Update ScenarioSpecs,ModuleID:185,Dwight.Sutton,Analysis,Phase2
2013-11-11,New Presentation(UI)-Logic,ModuleID:14 From (New ScenarioSpecs(1)),Donald.Reyes,Coding,Phase2
2013-11-12,Update AutomatedTesting(UFT),ModuleID:14 From (New ScenarioSpecs(1)),Eugene.Casey,Testing,Phase2
2013-11-13,Update Business-Logic,ModuleID:15 From (New ScenarioSpecs(0)),Levi.Wolfe,Coding,Phase2
2013-11-14,New ManualTesting,ModuleID:15 From (New ScenarioSpecs(0)),Wilfred.Owens,Testing,Phase2
2013-11-15,New Presentation(UI)-Logic,ModuleID:16 From (New ScenarioSpecs(1)),Terence.Hicks,Coding,Phase2
2013-11-15,Update ManualTesting,ModuleID:16 From (New ScenarioSpecs(1)),Benny.Ramos,Testing,Phase2
```

VI. COMPARISON OF BAYESIAN AND SUBSPACE ALGORITHMS

- The Annual dataset is sent as the input to the Bayesian and subspace algorithms.
- After the implementation of algorithms, the outputs of each algorithm were compared using the cost factor.
- Based on the cost the decision is made whether to develop the next version or to resolve the defects in the present version.
- It is observed that the subspace algorithm is giving the best results and is shown in figure 4

.

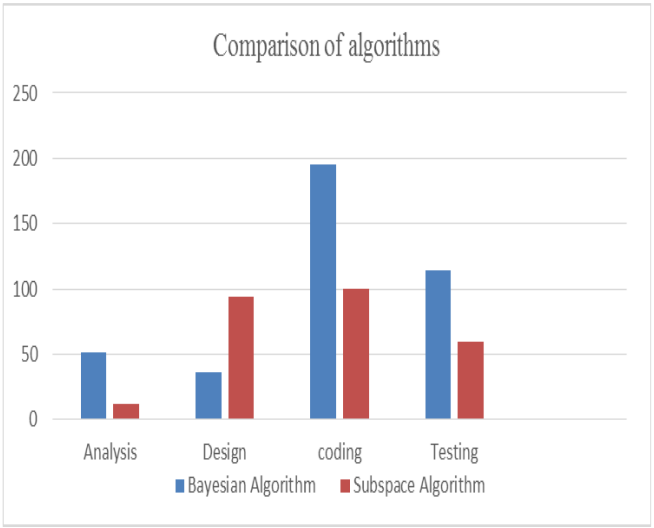

Fig 4: Comparison of Bayesian and Subspace algorithms

VII. CONCLUSION AND FUTURE WORK

Software defect and cost prediction is a technique to develop a software model in order to predict the defects in software modules. This paper evaluated the using of machine learning algorithms in software cost and defect prediction problem. Two machine learning techniques have been used, which are Bayesian and Subspace algorithms. The algorithm are also compared by applying on project data set at various phases and the comparison results shown that subspace algorithm is giving best results. Bayesian algorithm is more practical in predictions but cannot handle the data if it is increasing gradually. The application of subspace algorithm overcomes the problem of data. By getting the cost and defect predictions, the decision can be made whether to start the project or not. If defects cannot be solved even at maintenance phase, then it may be better not to start a project. As a future work, it may be extended involving other Machine Learning techniques and can be compared.

REFERENCES

1. http://www.thefullwiki.org/Software_development_methodology
2. Prasad Reddy P.V.G.D, Sudha K.R, Rama Sree P and Ramesh S.N.S.V.S.C, (2010), "Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks", Journal of Computing, Volume 2, Issue 5, pp 87-92.
3. Park, H., Baek, S.: An empirical validation of a neural network model for software effort estimation. Expert System with Applications 35,929-937 (2008).
4. https://www.kdnuggets.com/2016/12/data-science-basics-types-patterns-mined-data.html
5. Martin Shepperd and Chris Schofield "Estimating Software Project Effort Using Analogies"
6. WanJiang Han1 , LiXin Jiang2 , TianBo Lu1 and XiaoYan Zhang "Comparison of Machine Learning Algorithms for Software Project Time Prediction"
7. Sefa Oguz Saglam "Applied Machine Learning: Project Management Performance Prediction At Information Technology Company Project Management Office"
8. B. Boehm, "Requirements that handle IKIWISI, COTS, and rapid change," IEEE Computer, Vol. 33, No. 7, July 2000, pp.99-102.
9. D. Parnas, "Designing software for ease of extension and contraction," IEEE Trans. SE, Vol. 5, No. 3, March 1979, pp. 128-137.
10. C. Green et al, "Report on a knowledge-based software assistant, In Readings in Artificial Intelligence and Software Engineering, eds. C. Rich and R.C. Waters, Morgan Kaufmann, 1986, pp.377-428.
11. M. Lowry, "Software engineering in the twenty first century", AI Magazine, Vol.14,No.3, Fall 1992, pp.71-87.
12. T. Mitchell, Machine Learning, McGraw-Hill, 1997.
13. Jaechang Nam "Survey on Software Defect Prediction"
14. Boetticher G, Menzies T, Ostrand T (2007) Promise repository of empirical software engineeringdatahttp://promisedata.org/ repository. Department of Computer Science, West Virginia University
15. N. Fenton and M. Neil, "A critique of software defect prediction models," IEEE Trans. SE, Vol. 25, No. 5, Sept. 1999, pp. 675-689
16. https://help.rallydev.com/sizing-and-estimates
17. F.V. Jensen, An Introduction to Bayesian Networks, Springer, 1996.

AUTHORS PROFILE

Sita Kumari.Kotha Ph.D from Acharya Nagarjuna University, Working as Associate Professor in VR Siddhartha Engineering College, AP, India. Research Areas. Software Engineering, Data Mining. Published more than 30 research papers in reputed journals and conferences.

.

Suhasini. Sodagudi Ph.D from Acharya Nagarjuna University, Working as Associate Professor in VR Siddhartha Engineering College, AP, India. Research Areas Network Security, Data Mining, . Published more than 30 research papers in reputed journals and conferences.

Anuradha. T Ph.D from Acharya Nagarjuna University, Working as professor in VR Siddhartha Engineering College, AP, India. Research Areas Data Mining and Machine Learning. Published more than 40 research papers in reputed journals and conferences.