VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590018

Project Report
On

"SOFTWARE BUG DETECTION USING MACHINE LEARNING "

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF DEGREE

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

SUBMITTED BY

Shree Nidhi Kumar (1JB19IS096)

Under the Guidance of

Mrs. Pavithra
Assistant,
Professor,
Dept. of ISE, SJBIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

SJB INSTITUTE OF TECHNOLOGY

BGS HEALTH AND EDUCATION CITY,
KENGERI, BENGALURU-560060, KARNATAKA, INDIA.

2022 - 2023

|| Jai Sri Gurudev ||
Sri Adichunchanagiri Shikshana Trust ®

SJB INSTITUTE OF TECHNOLOGY

BGS Health & Education City, Kengeri, Bengaluru – 560 060

Department of Information Science & Engineering

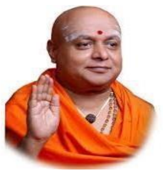
CERTIFICATE

Certified that the project work entitled **"SOFTWARE BUG DETECTION USING MACHINE LEARNING"** carried out by **Mr. Shree Nidhi Kumar bearing USN 1JB19IS096,** is bonafide student of SJB Institute of Technology in partial fulfillment for the award of BACHELOR OF ENGINEERING in INFORMATION SCIENCE AND ENGINEERING of the Visvesvaraya Technological University, Belagavi during the Academic Year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report (Phase -2)has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Mrs. Pavithra **Assistant, professor** **Dept. of ISE**	**Dr. Shashidhara H R** **Professor & Head** **Dept. of ISE**	**Dr. K V Mahendra Prashanth** **Principal** **SJBIT, Bengaluru**

External Viva

Name of the Examiners	**Signature with date**
1. ……………………………….	**…………………………..**
2. ……………………………….	**…………………………..**



ACKNOWLEDGEMENT

We would like to express our profound grateful to His Divine Soul **Padmabhushan Sri Sri Sri Dr. Balagangadharanatha Maha Swamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Maha Swamiji** for providing us an opportunity to complete our academics in this esteemed institution.

We would also like to express our profound thanks to **Revered Sri Sri Dr. Prakashnath Swamiji**, Managing Director, SJB Institute of Technology, for his continuous support in providing amenities to carry out this project in this admired institution.

We express our gratitude to **Dr. K V Mahendra Prashanth**, Principal, SJB Institute of Technology, for providing us an excellent facilities and academic ambience; which have helped usin satisfactory completion of project work.

We extend our sincere thanks to **Dr. Shashidhara H R**, Professor & Head, Department of Information Science and Engineering; for providing us an invaluable support throughout the period of our project work.

We wish to express our heartfelt gratitude to our guid**e, Mrs. Pavithra**, Assistant Professor**,** Department of Information Science and Engineering and the project coordinator **Dr. Rekha B,** Professor & Head Computer Science and Engineering (Data Science), for their valuable guidance, suggestions and cheerful encouragement during the entire period of our project work.

Finally, we take this opportunity to extend our earnest gratitude and respect to our parents, teaching & non-teaching staff of the department, and all our friends, who have directly orindirectly supported us during the period of our phase -1 project work.

Regards,
Shree Nidhi Kumar (1JB19IS096)

ABSTRACT

In the software bug detection it is basically in the software Industry while development there is a process of Software Engineering like waterfall model, spiral model, incremental model. These are the processes followed in the Information Technology Industries. One of the methodologies of SDLC (software development life cycle) is waterfall model but it has some drawbacks so, most of the IT company don't use this model because at the end of the processes by using this waterfall model they cannot fix all the generated bugs at the end process of the software development. Most of the IT companies use the Incremental model methodology such as agile and scrum methodology. In that the developers coded by using various programming languages. While coding different bugs generated and these bugs are actually like it would be written in the lock files and the lock files are generated every time and the bugs' generated every time. So for that we can apply Machine Learning Algorithm and Natural Language Processing to predict whether there are errors of bugs or not by using the text messages. So by using the lock files taking the inputs and on the basis of these inputs we can find whether there are bugs in the code or not. So, we need to identify that whether there are minor bugs or major bugs.

The big challenges of the IT industry to fix these complicated bugs because at the time of delivery of these system software where they use all the following prototypes before giving the system software to their clients and if there are still bugs then there may be higher chances to lose their clients and having a negative impressions and hence they immediately get the bad feedback given from their clients. So, we in order to overcome these problems we need to compose a noble methodology which can automatically fix all these complicated bugs by using NLP which can predict whether there is bugs or not. So, we need to integrate both the NLP and ML Algorithms. This is unstructured data and to the unstructured data we will apply the Natural Language Processing and Machine Learning Algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

The software development process is a structured process. It also refers to the Software Development Life Cycle (SDLC) methodology in which it involves many phases: Requirement analysis, design, testing, implementation & maintenance. SDLC is exposed to defects; non-detected faults occur after production. These defects could be due to a syntax error, a spelling error, an incorrect program declaration, requirements, design errors or specializations. Defects in software can make system error in which substantial lossesoccur in the event of a product launch, not acceptable. In today's world the number ofsoftware applications being developed is increasing day by day. Building highly reliable software requires a considerable amount of testing and debugging. As a result, software defect prediction techniques, which predict the occurrences of bugs, have been widely usedto assist developers in prioritizing their testing and debugging efforts. Predictions resultscan assist developers in prioritizing their testing and debugging eorts. SDLC is exposed todefects; non-detected faults occur after production. These defects could be due to a syntaxerror, a spelling error, an incorrect program declaration, requirements, design errors or specializations. Defects in software can make system error in which substantial losses occur in the event of a product launch, not acceptable.

In today's world the number of software applications being developed is increasing day by day. Building highly reliable software requires a considerable amount of testing and debugging. As a result, software defect prediction techniques, which predict the occurrences of bugs, have been widely used to assist developers in prioritizing their testing and debugging efforts. Predictions results can assist developers in prioritizing their testing and debugging. SDLC is exposed to defects; non-detected faults occur after production. These defects could be due to a syntax error, a spelling error, an incorrect program declaration, requirements, design errors or specializations. Defects in software can make system error in which substantial losses occur in the event of a product launch, not acceptable

In the software Bug detection it is basically in software industry while development there is a process of Software Engineering like waterfall model, spiral model, incremental model. These are the process followed in Information Technology Industries. One of the methodologies of SDLC (software development life cycle)is a waterfall model but it has some drawbacks, so most IT Company don't use the waterfall model because at the end of the processes by this waterfall model they cannot fix all generated bugs at the end process of software development. Most of The IT Companies uses Incremental model methodology such as agile and scrum methodology.In that developers coded by using various programming languages. While coding different bugs are generated and this bugs are actually like it would be written in log file and the log files are generated every time and the bug's generated every time.

The programmers get the bugs messages or warning messages they were looking to fix these bugs. If there is a less number of a code then it is easy to fix these bugs but if there is a large no of codes then there are a more chances of generating bugs so that it is very challenging job in IT Industry to fix these bugs. Bug reports are essential in helping developers triage, replicate, locate, and fix the bugs in the system's software. From the information that reporters provide in bug reports, the system's observed (unexpected) behavior (OB), the steps to reproduce the bug (S2R), and the software expected behavior (EB) are among the most important elements for developers. These elements are typically expressed by end-users or developers in free-form natural language through issue tracker. While these elements are essential, they are often incomplete, unclear, or not provided at all by the reporters .The consequence of this is that developers often spend too much effort triaging and fixing the problems, and often, they cannot even reproduce and fix the bugs inthe code .One of the main reasons for having low-quality bug reports is the lack of feedback and quality verification of issue trackers.

1.1 Software defect classification

Different types of software defects which can originate in various phases of SDLC are as follows :

- **Requirements defects:** These defects are due to poor usage of wordings and problem in organization of document which makes difficult to understand.

- **Design defects:** Design defects can be design error or omission. Design error can due to lack in design element construction. Design omission will be missing of design items while construction of design.

- **Computational defects:** These defects are mainly because of incorrect usage of equations or data structure within the source code which will lead to miscalculation.

- **Logical defects:** Incorrect conditional statements in looping structure of software modules. Irrelevant operators'usage in conditional statements.

- **Interface defects:** May be due incorrect handling of passing parameters in methods. Interface defects can internal or external.

- **Database defect:** Error in schema design of database.

- **Testing defect:** Failure to identify and report the problem in software modules.

LITERATURE SURVEY

Title	Year	Tools	Application Area	Pros	Cons	Analysis
Towards Effective Trouble shooting With Data Truncation [1]	2015	Instance selection and feature selection	Bug Repository Handling.	The problem of handling huge number of data in bug repository is minimized	Instance selection and feature selection is not completely enough to handle the data in bug repository	Additionally, text processing and data processing algorithm must be used to handle the data in bug repository.
A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage [2]	2015	Instance selection and feature selection	For an efficient bug triage.	It analysis data by considering the word dimension and bug dimension which helps in reducing duplicate and unnecessary bugs.	The order of applying instance selection and feature selection is not clearly explained which leads to inefficient system.	The order of applying instance selection and feature is selection must be determined by a predictive Algorithm.
Automatic Bug Triage using Semi-SupervisedText Classification [3]	2010	Naïve bayes classifier	Software bug handling.	It labels the bug data iteratively. The weighted list maintained, helps to boost theresults obtained.	It only focuses on classifying the bugs inbug repository. The major problem in bug handling is that huge number ofdata in bug repository.	Techniques such as IS and FS must be used to reduce the data in bug repository. Proper predictive algorithm must beused to classify bugs.
Approach of Data Reduction Techniques for Bug Triaging System [4]	2008	selection and historical data sets		reduce the data in bug repository. Then uses predictive model to predict a developer based on historical	the bugs in bug repository which can be minimized.	to comparethe new bugreport with historical data sets.

A survey on bug-report analysis [4]	**2008**	**Bugzilla**	**Explains various techniquesin bughandling**	**Research and explain background details aboutbug handling which makes bug handling an efficient task.**	**The accuracy obtains by this techniques arenot more enough in handling thebugs.**	**Several more techniques such as instance selection and feature selection must be usedto handle the bugs.**
Automatic bug triage using text Categorization [5]	2007	Naïve bayes classifier	Large open source projects.	Have explained naïve bayes classifier forbug triaging. And investigatedbug data sets efficiently for bug handling.	There is a chance of assigningth e bugs to un- suited developers.	Developers should be involved inbug triaging for an efficient bug handling.
Automatic bug triage using text categorization [6]	2004	Software development community		It focuses on Developing a model that provides enhanced software management techniques to handle bugs.	It just bugs from single community.	Tools must be int ro d u c e d t o handle BRN's complexity.
They used NASA MDP and maximum accurac was 0.857 for data using ANFS me	**2019**	**performance measure using RaF, KNN and MLP -accuracy 90.765%**	**Preprocessing,& Hybrid Logistic Regression (LR) Techniques**	**erformance measure using RaF, N and MLP - accuracy 90.765%**	**- aF, KNN, MLP -1 -fold cross v alidation test ode - KA tool used**	**-To predict software fault using Data Preprocessing,& Hybrid Logistic Regression (LR) Techniques**

PROBLEM STATEMENT

The primary goal of the proposed study is to evolve up with a technique that can perform automatic prediction of software defects from large, complex, and highly unstructured data. In order to accomplish the above mentioned goal, following objectives are targeted to me. To review various literatures for understanding the effectiveness of the prior techniques presented and evolve up with a significant research gap and open issues. Data cleaning, Pre-processing and extract useful text from data To design a new framework which can combine different parameters for feature selection to give more accurate prediction. Applying the classification algorithms such as random forest for software defects classification. Predict andmeasure the performance the various machine learning algorithm.

In the software Bug detection it is basically in software industry while development there is a process of Software Engineering like waterfall model, spiral model, incremental model. These are the process followed in Information Technology Industries. One of the methodologies of SDLC(software development life cycle)is a waterfall model but it has some drawbacks, so most IT Company don’t use the waterfall model because at the end of the processes by this waterfall model they cannot fix all generated bugs at the end process of software development. Most of The IT Companies uses Incremental model methodology suchas agile and scrum methodology. In that developers coded by using various programming languages. While coding different bugs are generated and this bugs are actually like it would be written in log file and the log files are generated every time and the bug’s generated every time.

CHALLENGES

Many embedded systems have substantially different designs according to their functions and utilities. Obstacle avoidance is one of the most important aspects of mobile robotics. Without it robot movement would be very restrictive and fragile. So,in order to overcome these problems, there are two important criteria that must be satisfied by this obstacle avoidance robot car. First, it should perform trajectory planning effectively by detecting the obstacles and moving in pre- computed path. Secondly, it is needed to risk regarding the upper limit of a human eye. Most of the IT companies use the Incremental model methodology such as agile and Scrum methodology. In that the developers coded by using various programming different bugs generated and these bugs are actually like it would be written in the lock files and the lock files are generated every time and the bugs' generated every time.The programmers get the bugs messages or warning messages they were looking to fix these bugs. If there is less number of codes then it is easily to fix these bugs but if there is large number of codes then there is more chance of generation of bugs so at that it is difficult to fix these bugs

So for that we can apply Machine Learning Algorithm and Natural Language Processing to predict whether there are errors of bugs or not by using the text messages. So by using the lock files taking the inputs and on the basis of these inputs we can find whether there are bugs in the code or not. So, we need to identify that whether there are minor bugs or major bugs. We are going to apply the NLP (Natural Processing Language) and Machine Learning Algorithms to predict whether there are defect in the processed code or not by using the text messages. This is a big complex thing because here some sentence of the output we are getting in the lock files.

MOTIVATION

It's hard to develop flawless software products and project managers, and software engineers are making a lot of work to curtail defects. If a defect is identified subsequently the delivery of the software product, the cost of the project low-priced as well as resources will be higher than the early detection cost. If there is less number of codes thenit is easily to fix these bugs but if there is large number of codes then there is more chanceof generation of bugs so at that it is difficult to fix these bugs. The software repository, code base, traces of execution, and comprises historical data, such as a change to the codehistory, the status of the software project, progress, comprises a wealth of info about the evolution.

OBJECTIVES

As the requirements increase, the IT industry has to do a lot of work to meet customer needs. As a result, data mining technology has been widely used to extract useful insights from large software repositories and accept it to detect bugs and quality of the software is improved. This project adapted the methods of detection and classification of software defects by incorporating machine learning techniques to identify and classify defects fromlarge software repositories. The activities described here use software defect metrics based on the severity of defects in software products for classification. Random forests get significantly better results than the other two classifications, KNN and Naive Bayes. Software development is inevitably subject to defects, and accurate prediction of these defects is the holy grail of most software development entities. The cost of finding and rectifying software defects is prohibitively expensive. Therefore, a lot of research effort is devoted to finding efficient methods that can reliably predict defective software modules. However, many of thestudies conducted to predict faulty modules in the software development process apply historical software defect data mined from online depositories. When defect prediction is considered as a binary classification problem in machine learning domain, prediction model developed with software metrics (i.e., features) identifies new software modules as a possibledefect or not . However, datasets obtained from the depositories come with high dimensionalfeature space, many of which are irrelevant and redundant. Irrelevant features consist of those that offer no useful information for the classification task, and redundant features present the same information as the currently selected features. Model built on training data with irrelevant and redundant features causes performance deterioration, increases model runtime and complexity

Chapter 2

DESIGN AND ARCHITECTURE

Software Bug Detection is an important issue in Software Development Life Cycle which concern about overall software successes. This is because predicting the software faults in earlier phase improves the software quality, reliability, efficiency and reduces the software cost. Basically Three Supervised ML Algorithms have been used to predict the future software faults based on the Historical data. This Classifier is Naive Bayes (NB), Decision Tree (DT), and Random Forest Algorithms.

In this project the first step is to do Data Preprocessing which is the major things in this project ,Basically here We used different feature Engineering techinique to make optimised dataset basically this include Feature Selection,Data Imbalance techique and some Hyperparameter Optimisation techinique to improves the accuracy by tunning some parameters in the Random Forest Algorithm, After we get the optimised dataset we will used to comapare the Accuracy with other Algorithms And gonna to check the Overall Accuracy with the Other Agorithms like Bagging Algorithms,MLP,RBF And other Algorithms. Finally we plot a Bar Graph to see the overall Performance of all the Algorithms.So In the next page we have shown the Proposed Archiecture will works.

- The First Step is to perform the Data Preprocessing part where we used Feature Engineering techiniques to convert raw data into optimised data.where we first checkthe Null value in the dataset then we used feature selection techinique where we used Heatmap visualization to find out most important features by dropping irrevelent features.
- The next step is to perform Data Imbalance techinique where where the number of observations per class is not equally distributed so we used SMOTE Techinique to make equally distributed within two classes.

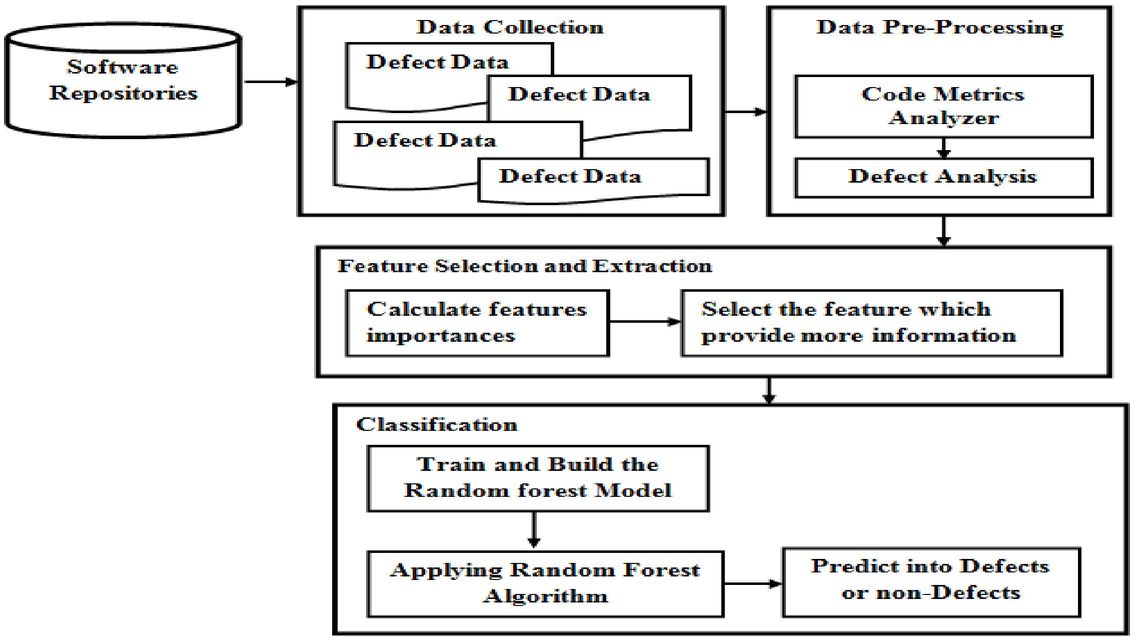

Software Repositories
Data Collection
Defect Data
Defect Data
Defect Data
Defect Data
Data Pre-Processing
Code Metrics Analyzer
Defect Analysis
Feature Selection and Extraction
Calculate features importances
Select the feature which provide more information
Classification
Train and Build the Random forest Model
Applying Random Forest Algorithm
Predict into Defects or non-Defects

Figure: 1 Proposed Architecture

- After performing all this we again perform Hyperparameter optimisation to improves the Accuracy of our supervised Machine Learning Algorithms by using Grid Search CV & Randomized Search CV Techiniques.
- In this project at last step we introduced various machine learning Algorithms and try to comapre the Accuracy with other Algorithms and going to check the overall Accuracy in Bar Graph.

- Basically in our dataset there were various unnecessary coloumns are there which are of no used so we need to remove those unnecessary features so we used the techinique called Heatmap Techinique to remove unnecessary coloumns from our dataset.

- Correlation states how the features are related to each other or the target variable.Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)

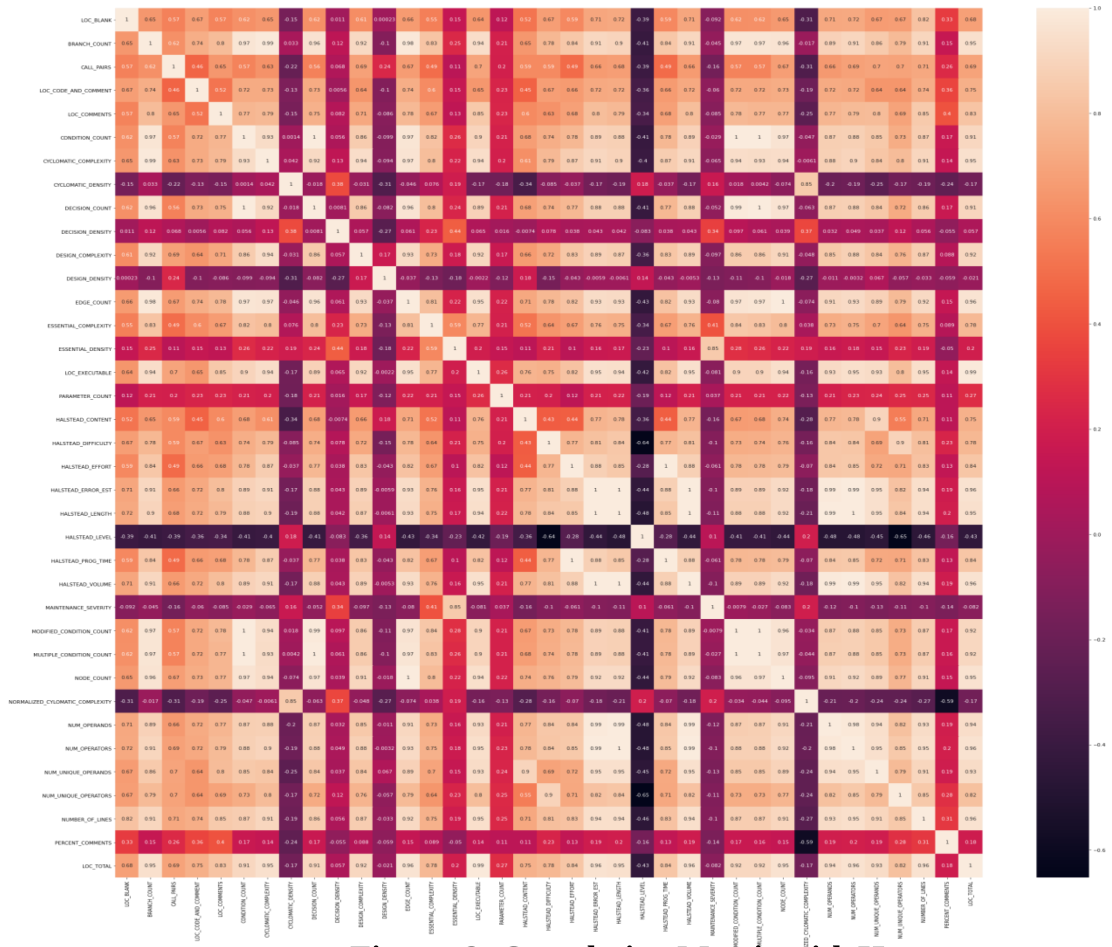


Figure: 2: Correlation Matrix with Heatmap

- Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.
- In the Above figure we used Heatmap Techinique to find most correlated features those features which are more correlated so used to drop any one of those features to remove unnecessary features.

- Basically the next step were we used data imbalance techinique of SMOTE Techinique it works by Find its k nearest neighbors (k_neighbors is specified as an argument in the SMOTE() function) Choose one of these neighbors and place a synthetic point anywhere on the line joining the point under consideration and its chosen neighbor. we repeat this step untill our data becomes completly balanced.

Naive Bayes algorithm

The Naive Bayes algorithm is an algorithm that learns the probabilities of objects with characteristics appropriate to a specific group / class. It is a probabilistic classifier. It creates the hypothesis of a feature which is independent of the appearance of other features, so it is called "naive". For example, if you try to identify fruits based on fruitcolor, shape and taste, the orange spherical fruits are probably orange. All these characteristics contribute individually to the probability that this fruit is orange, and therefore are known as "naive".

The Mathematics of the Naive Bayes Algorithm.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

- $P(A|B)$: **Probability (conditional probability) of occurrence of event A given the event B is true**
- $P(A)$ **and** $P(B)$: **Probabilities of the occurrence of event A and B respectively**
- $P(B|A)$: **Probability of the occurrence of event B given the event A is true**

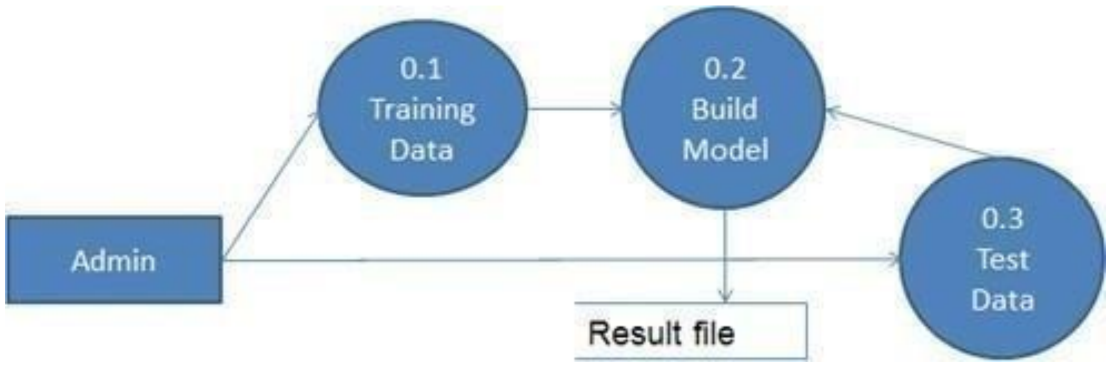


Figure: 3: DATAFLOW DIAGRAM LEVEL-1

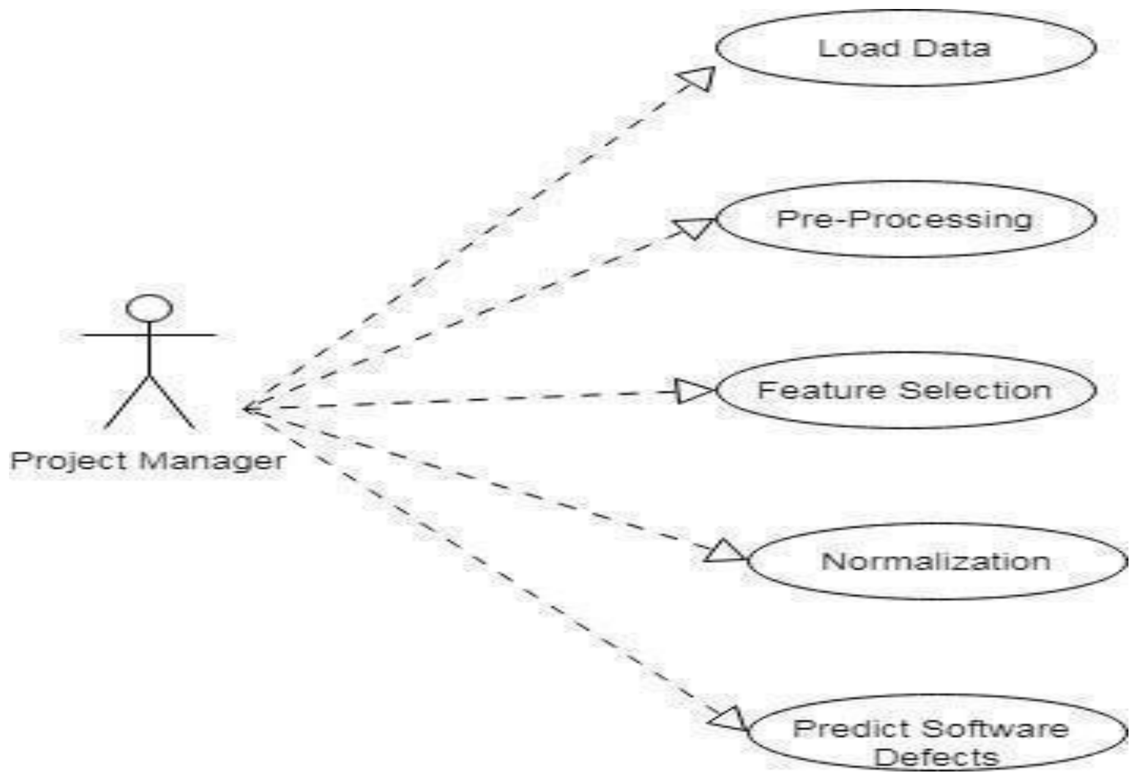


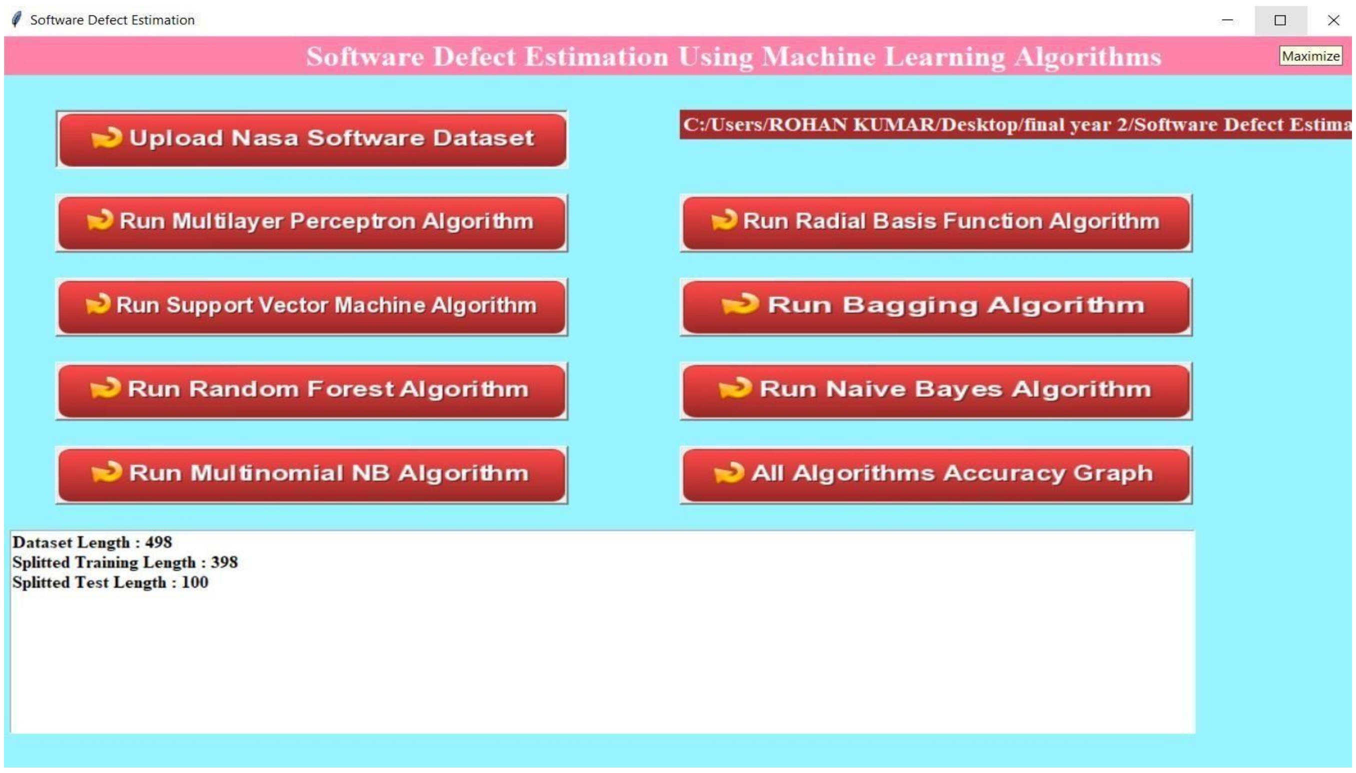
Figure 4. USE CASE DIAGRAM OF PROPOSED SYSTEM

Modules Used For Implementation

- Data acquisition
- Exploratory data analysis
- Data pre-processing
    - Feature selection.
    - Feature scaling.
- Modelcreation
    - Split data for train and test.
    - Hypothesis and modeling using random forest.
    - Evaluation and interpretation using test data.
- Predict software defect for new dataset.

- We have many algorithms Like MLP, SVM, Multinomial NB ,RBF, Bagging and Naive Bayes Algorithms.

WORKING


Software Defect Estimation
Maximize
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
C:/Users/ROHAN KUMAR/Desktop/final year 2/Software Defect Estima
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Dataset Length : 498
Splitted Training Length : 398
Splitted Test Length : 100
```

Figure 5: Working Classification Algorithm

- But among these which will give the best accuracy that will be chosen so here we have the optimised dataset already in the txt format which we have got after performming Data pre-processing Techiniques.

- After then we upload our dataset by clicking in Upload Nasa Software Dataset button after then we run all the machine learning algorithms over that dataset and check the accuracy.

- In Below area there is an Screen area where it is showing training and testing splitting and the accuracy of various Algorithms.

Chapter 3

METHODOLOGY

- **Collection of Dataset**

  In the first step we collected our dataset by visiting various research papers. From there we have taken our dataset which contains various features including Loc Blank, Branch Count, Cyclometric Complexity, Halstead Effort, Num Unique Operands etc.

- **Data Preprocessing**

  Since our dataset contains various bugs so we need to perform Data Preprocessing Principal to clean our dataset, which included various Feature Engineering Techniques like Heat map Correlation Matrix with Heat map then we used Data imbalanced techniques (SMOTE) this help us to make our dataset more optimised by improving the accuracy.

- **Apply Machine Learning Algorithms**

  This is one of the most important step in our project we are applying train test split upon our optimised dataset then we apply different machine learning algorithms to check which hypothesis is giving us more accuracy. and at last we show the graph upon that accuracy.

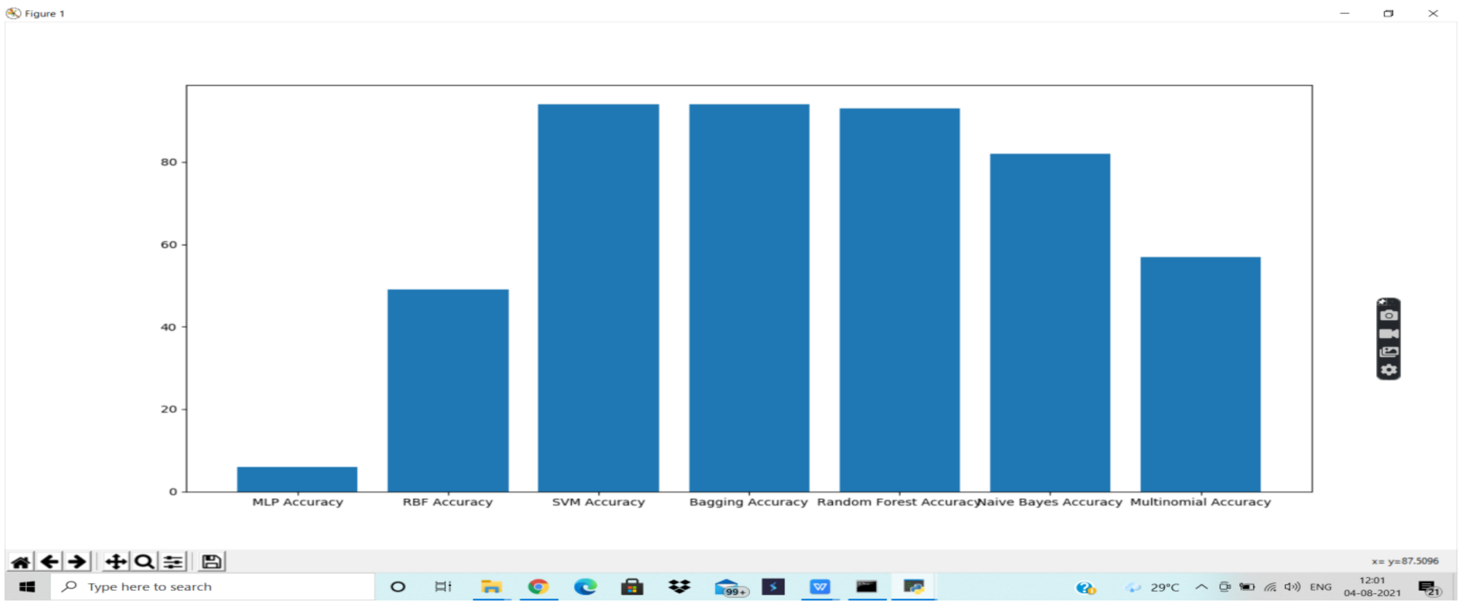


Figure 6: All Algorithms Graph Accuracy

STEPS TO IDENTIFY THE DEFECTSAND CLASSIFICATION

Step 1: Collect data series from various sources of defect data (eg Promise repository, NASA MDA repository)

Step 2: Measure the source code using static code metrics. The various metrics used are the following:

McCabe metrics. Halstead stress, Halstead effort, Halstead error estimate, Halstead length, Halstead level, Halstead programming time and Halstead amount, LOC metrics. Comment LOC, number of operands, number of unique operands and number of unique operators and finally defect metrics. Number of errors, density of the error, number of defects (including gravity and priority information).

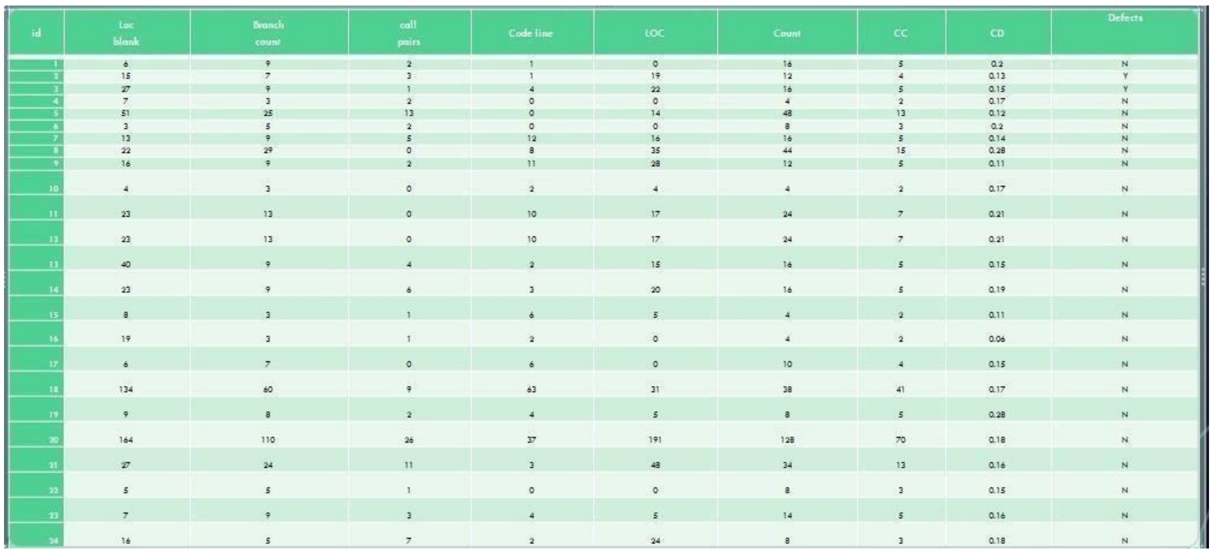
Step 3: Defect data is extracted from the software repository and stored in the database for preprocessing

Step 4: apply machine learning techniques to identify and classify the severity of defects

Step 5: Defect Classification

Step 6: Software defects prediction for new set of data.

Dataset collected from CMI software

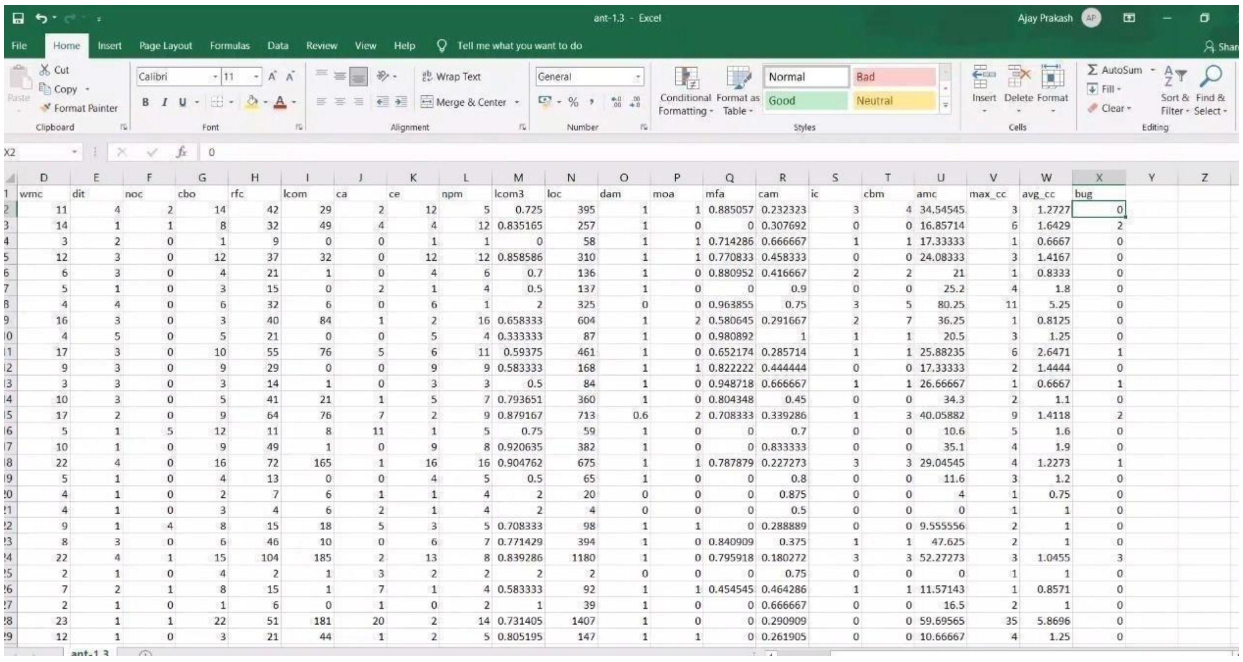


id	Loc blank	Branch count	call pairs	Code line	LOC	Count	CC	CD	Defects
1	6	9	2	1	0	16	5	0.2	N
2	15	7	3	1	19	12	4	0.13	Y
3	27	9	1	4	22	16	5	0.15	Y
4	7	3	2	0	0	4	2	0.17	N
5	51	25	13	0	14	48	13	0.12	N
6	3	5	2	0	0	8	3	0.2	N
7	13	9	5	12	16	16	5	0.14	N
8	22	29	0	8	35	44	15	0.28	N
9	16	9	2	11	28	12	5	0.11	N
10	4	3	0	2	4	4	2	0.17	N
11	23	13	0	10	17	24	7	0.21	N
12	23	13	0	10	17	24	7	0.21	N
13	40	9	4	2	15	16	5	0.15	N
14	23	9	6	3	20	16	5	0.19	N
15	8	3	1	6	5	4	2	0.11	N
16	19	3	1	2	0	4	2	0.06	N
17	6	7	0	6	0	10	4	0.15	N
18	134	60	9	63	31	38	41	0.17	N
19	9	8	2	4	5	8	5	0.28	N
20	164	110	26	37	191	128	70	0.18	N
21	27	24	11	3	48	34	13	0.16	N
22	5	5	1	0	0	8	3	0.15	N
23	7	9	3	4	5	14	5	0.16	N
24	16	5	7	2	24	8	3	0.18	N

Figure 7:Dataset Collected from CMI software

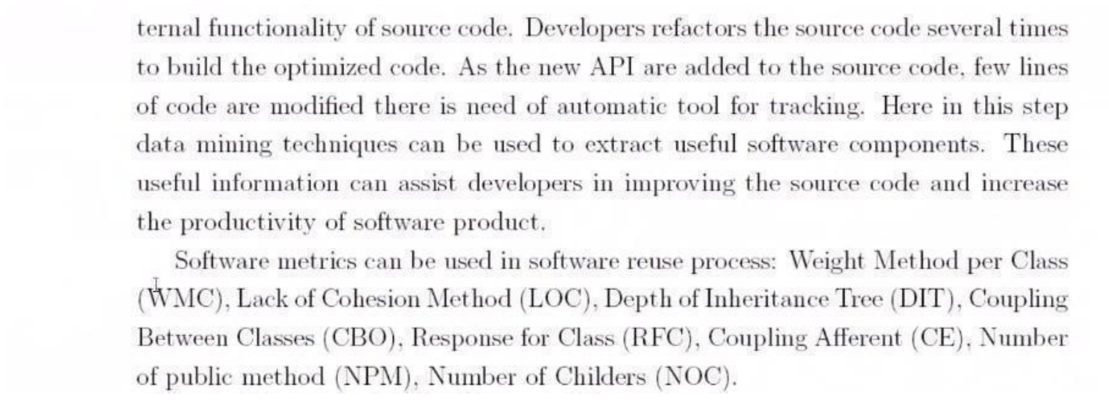
Chapter 4

RESULT



wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom3	loc	dam	moa	mfa	cam	ic	cbm	amc	max_cc	avg_cc	bug
11	4	2	14	42	29	2	12	5	0.725	395	1	1	0.885057	0.232323	3	4	34.54545	3	1.2727	0
14	1	1	8	32	49	4	4	12	0.835165	257	1	0	0	0.307692	0	0	16.85714	6	1.6429	2
3	2	0	1	9	0	0	1	1	0	58	1	1	0.714286	0.666667	1	1	17.33333	1	0.6667	0
12	3	0	12	37	32	0	12	12	0.858586	310	1	1	0.770833	0.458333	0	0	24.08333	3	1.4167	0
6	3	0	4	21	1	0	4	6	0.7	136	1	0	0.880952	0.416667	2	2	21	1	0.8333	0
5	1	0	3	15	0	2	1	4	0.5	137	1	0	0	0.9	0	0	25.2	4	1.8	0
4	4	0	6	32	6	0	6	1	2	325	0	0	0.963855	0.75	3	5	80.25	11	5.25	0
16	3	0	3	40	84	1	2	16	0.658333	604	1	2	0.580645	0.291667	2	7	36.25	1	0.8125	0
4	5	0	5	21	0	0	5	4	0.333333	87	1	0	0.980892	1	1	1	20.5	3	1.25	0
17	3	0	10	55	76	5	6	11	0.59375	461	1	0	0.652174	0.285714	1	1	25.88235	6	2.6471	1
9	3	0	9	29	0	0	9	9	0.583333	168	1	1	0.822222	0.444444	0	0	17.33333	2	1.4444	0
3	3	0	3	14	1	0	3	3	0.5	84	1	0	0.948718	0.666667	1	1	26.66667	1	0.6667	1
10	3	0	5	41	21	1	5	7	0.793651	360	1	0	0.804348	0.45	0	0	34.3	2	1.1	0
17	2	0	9	64	76	7	2	9	0.879167	713	0.6	2	0.708333	0.339286	1	3	40.05882	9	1.4118	2
5	1	5	12	11	8	11	1	5	0.75	59	1	0	0	0.7	0	0	10.6	5	1.6	0
10	1	0	9	49	1	0	9	8	0.920635	382	1	0	0	0.833333	0	0	35.1	4	1.9	0
22	4	0	16	72	165	1	16	16	0.904762	675	1	1	0.787879	0.227273	3	3	29.04545	4	1.2273	1
5	1	0	4	13	0	0	4	5	0.5	65	1	0	0	0.8	0	0	11.6	3	1.2	0
4	1	0	2	7	6	1	1	4	2	20	0	0	0	0.875	0	0	4	1	0.75	0
4	1	0	3	4	6	2	1	4	2	4	0	0	0	0.5	0	0	0	1	1	0
9	1	4	8	15	18	5	3	5	0.708333	98	1	1	0	0.288889	0	0	9.555556	2	1	0
8	3	0	6	46	10	0	6	7	0.771429	394	1	0	0.840909	0.375	1	1	47.625	2	1	0
22	4	1	15	104	185	2	13	8	0.839286	1180	1	0	0.795918	0.180272	3	3	52.27273	3	1.0455	3
2	1	0	4	2	1	3	2	2	2	2	0	0	0	0.75	0	0	0	1	1	0
7	2	1	8	15	1	7	1	4	0.583333	92	1	1	0.454545	0.464286	1	1	11.57143	1	0.8571	0
2	1	0	1	6	0	1	0	2	1	39	1	0	0	0.666667	0	0	16.5	2	1	0
23	1	1	22	51	181	20	2	14	0.731405	1407	1	0	0	0.290909	0	0	59.69565	35	5.8696	0
12	1	0	3	21	44	1	2	5	0.805195	147	1	1	0	0.261905	0	0	10.66667	4	1.25	0

Figure: 8 Software Metrics



ternal functionality of source code. Developers refactors the source code several times to build the optimized code. As the new API are added to the source code, few lines of code are modified there is need of automatic tool for tracking. Here in this step data mining techniques can be used to extract useful software components. These useful information can assist developers in improving the source code and increase the productivity of software product.

Software metrics can be used in software reuse process: Weight Method per Class (WMC), Lack of Cohesion Method (LOC), Depth of Inheritance Tree (DIT), Coupling Between Classes (CBO), Response for Class (RFC), Coupling Afferent (CE), Number of public method (NPM), Number of Childers (NOC).

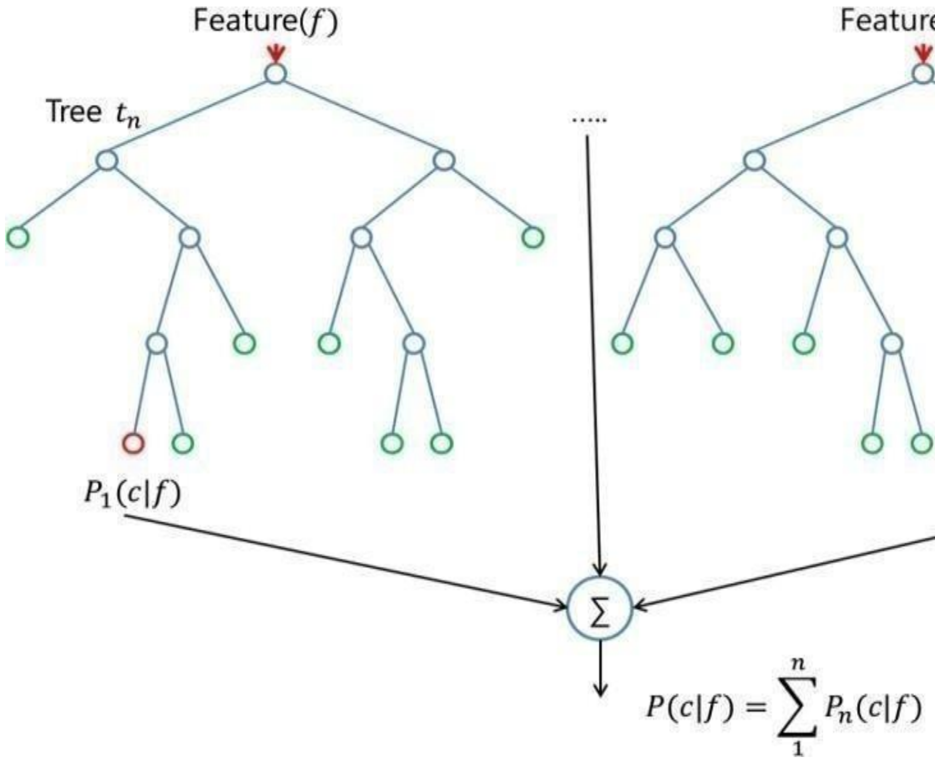
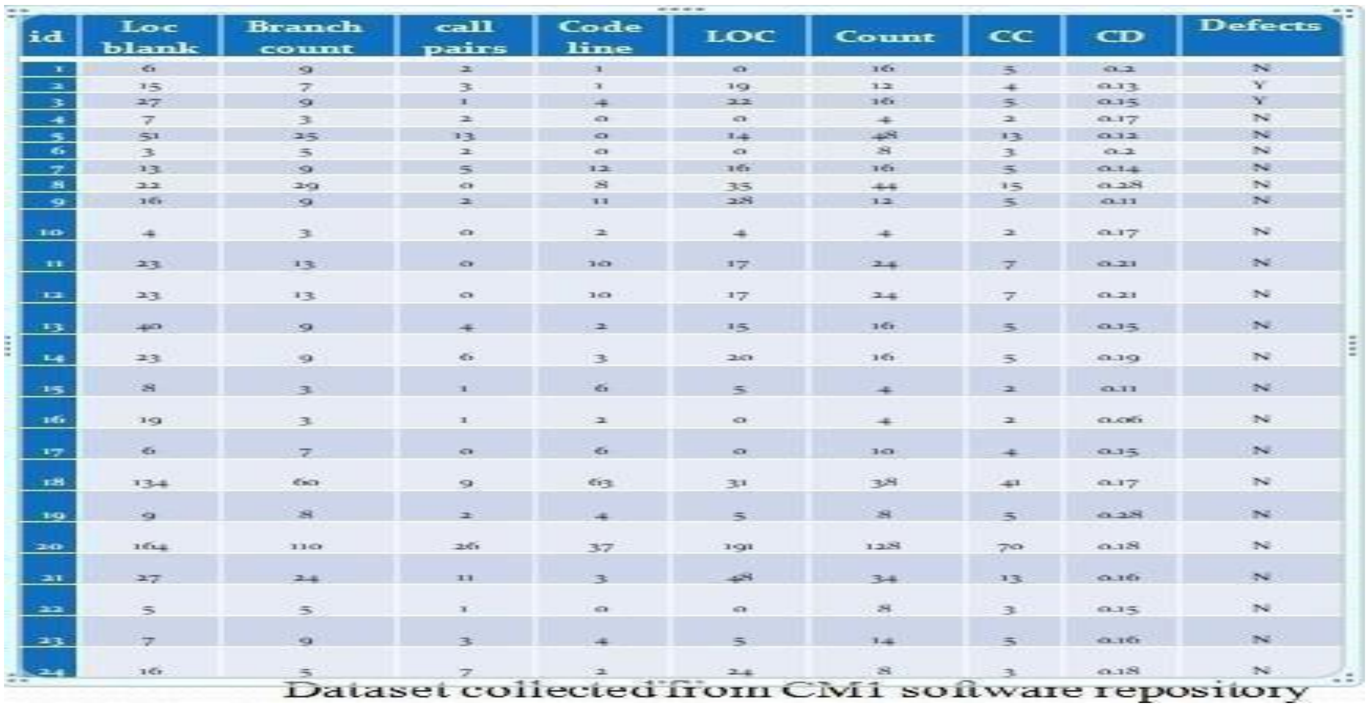


Figure: 9: Random Forest Structure

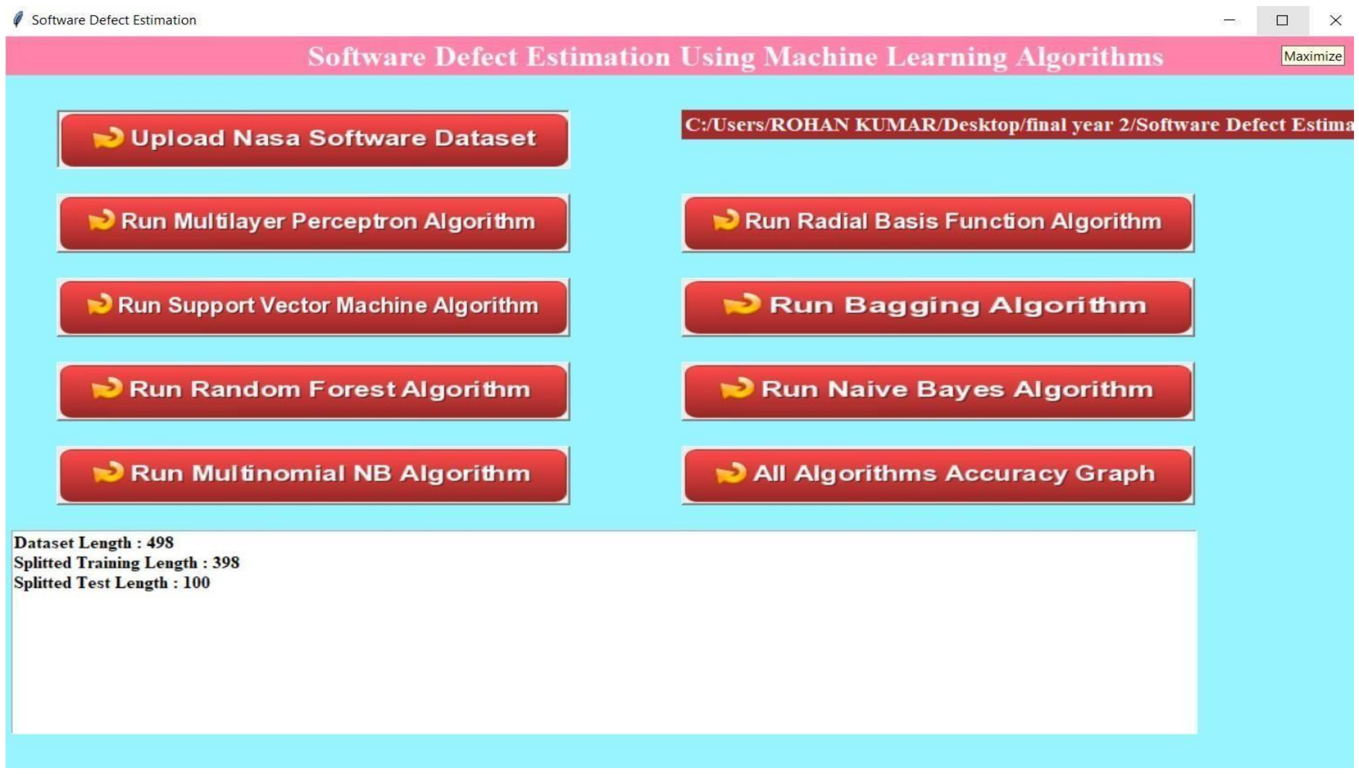
Random Forest structure



id	Loc blank	Branch count	call pairs	Code line	LOC	Count	CC	CD	Defects
1	6	9	2	1	0	16	5	0.2	N
2	15	7	3	1	19	12	4	0.13	Y
3	27	9	1	4	22	16	5	0.15	Y
4	7	3	2	0	0	4	2	0.17	N
5	51	25	13	0	14	48	13	0.12	N
6	3	5	2	0	0	8	3	0.2	N
7	13	9	5	12	16	16	5	0.14	N
8	22	29	0	8	35	44	15	0.28	N
9	16	9	2	11	28	12	5	0.11	N
10	4	3	0	2	4	4	2	0.17	N
11	23	13	0	10	17	24	7	0.21	N
12	23	13	0	10	17	24	7	0.21	N
13	40	9	4	2	15	16	5	0.15	N
14	23	9	6	3	20	16	5	0.19	N
15	8	3	1	6	5	4	2	0.11	N
16	19	3	1	2	0	4	2	0.06	N
17	6	7	0	6	0	10	4	0.15	N
18	134	60	9	63	31	38	41	0.17	N
19	9	8	2	4	5	8	5	0.28	N
20	164	110	26	37	191	128	70	0.18	N
21	27	24	11	3	48	34	13	0.16	N
22	5	5	1	0	0	8	3	0.15	N
23	7	9	3	4	5	14	5	0.16	N
24	16	5	7	2	24	8	3	0.18	N

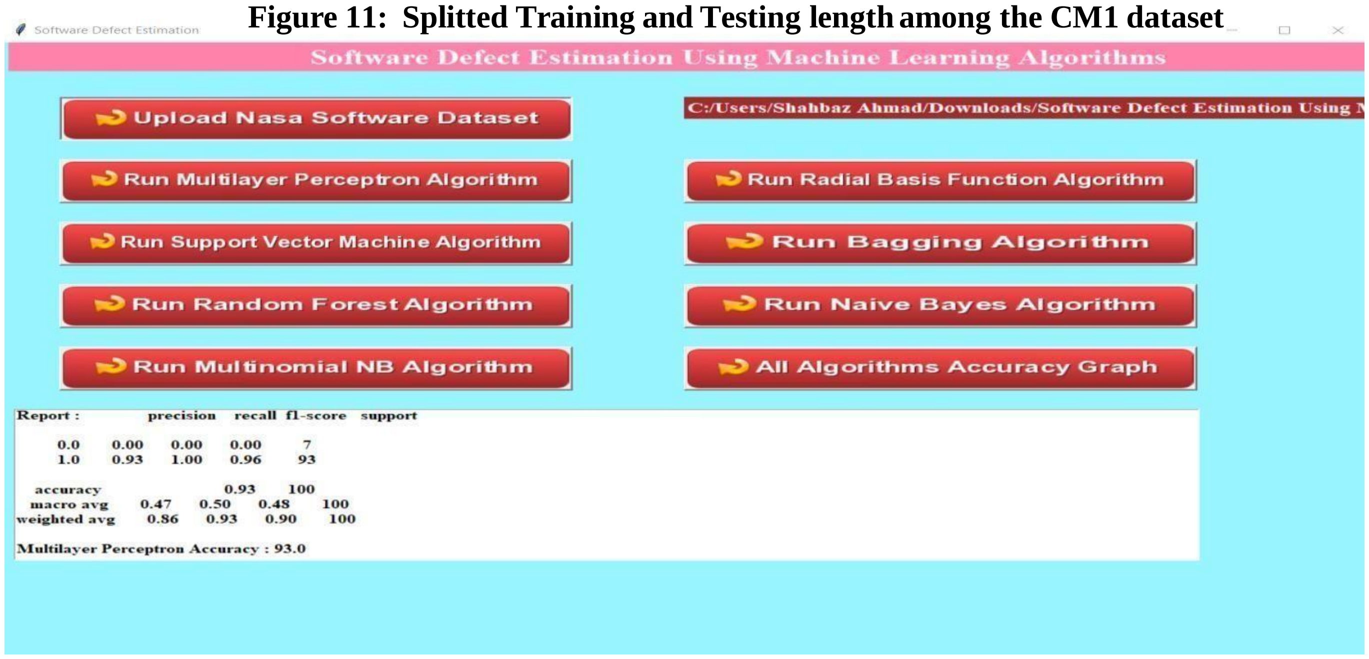
Dataset collected from CM1 software repository

Figure: 10: Dataset Collected from CMISoftware repository


Software Defect Estimation
Software Defect Estimation Using Machine Learning Algorithms
Maximize
Upload Nasa Software Dataset
C:/Users/ROHAN KUMAR/Desktop/final year 2/Software Defect Estima
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Dataset Length : 498
Splitted Training Length : 398
Splitted Test Length : 100
```

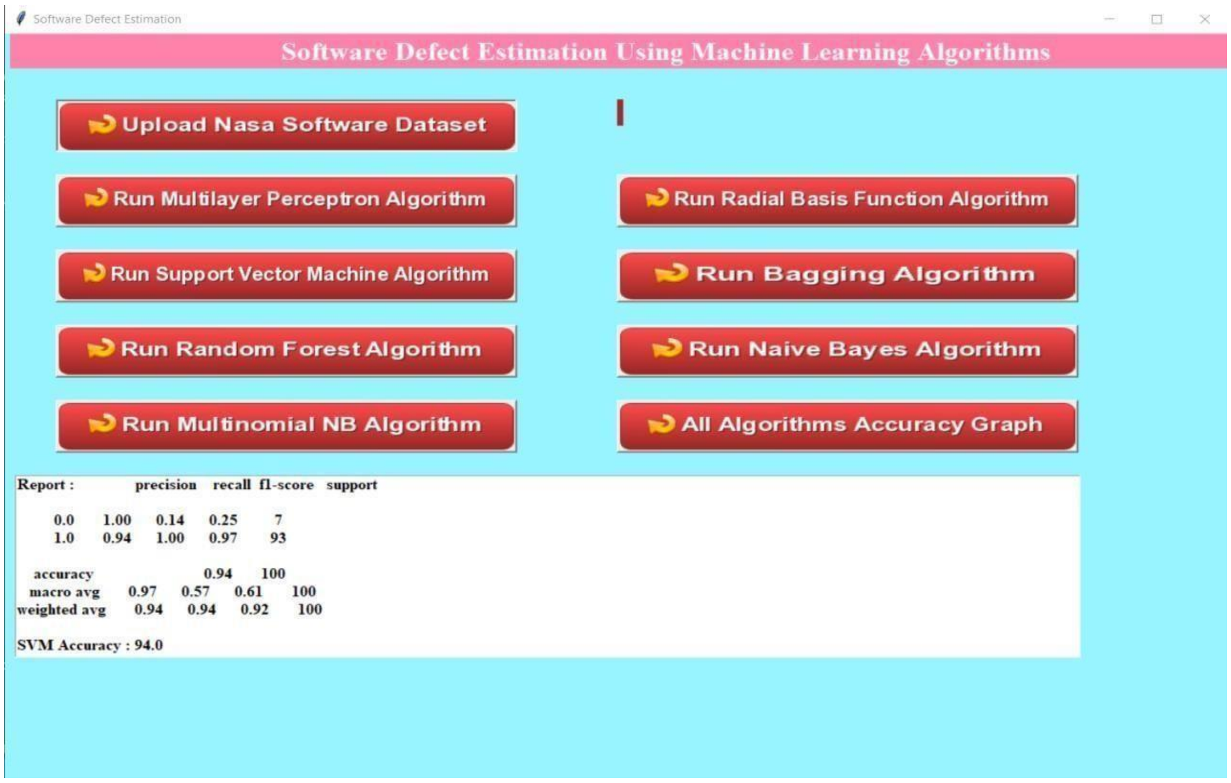
Figure 11: Splitted Training and Testing length among the CM1 dataset


Software Defect Estimation
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
C:/Users/Shahbaz Ahmad/Downloads/Software Defect Estimation Using M
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :          precision    recall  f1-score   support

     0.0       0.00      0.00      0.00         7
     1.0       0.93      1.00      0.96        93

    accuracy                           0.93       100
   macro avg       0.47      0.50      0.48       100
weighted avg       0.86      0.93      0.90       100

Multilayer Perceptron Accuracy : 93.0
```

Figure 12: Accuracy of Multilayer Perceptron using CM1 dataset : 93.0

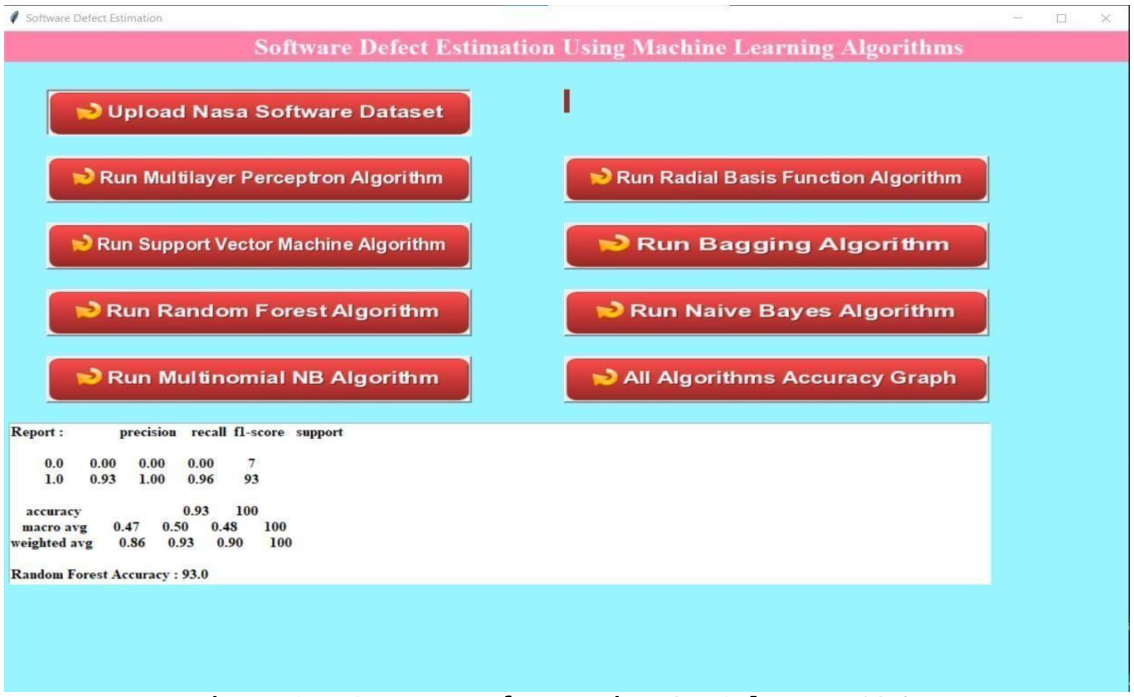
[i] **Multilayer Perceptron**: Multilayer Perceptron which is one of the types of Neural Networks comprises of one input layer, one output layer and at least one or more hidden layers. This algorithm transfers the data from the input layer to the output layer, which is called feed forward. For training, the back propagation technique is used. One hidden layer with (attributes + classes) / 2 units are used for this experiment. Each dataset has 22 attributes and 2 classes which are false and true. We determined the learning rate as 0.3 and momentum as 0.2 for each dataset.


Software Defect Estimation
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :          precision    recall  f1-score   support

      0.0       1.00      0.14      0.25         7
      1.0       0.94      1.00      0.97        93

    accuracy                           0.94       100
   macro avg       0.97      0.57      0.61       100
weighted avg       0.94      0.94      0.92       100

SVM Accuracy : 94.0
```

Figure 13: Accuracy of SVM using CM1 dataset: 94.0

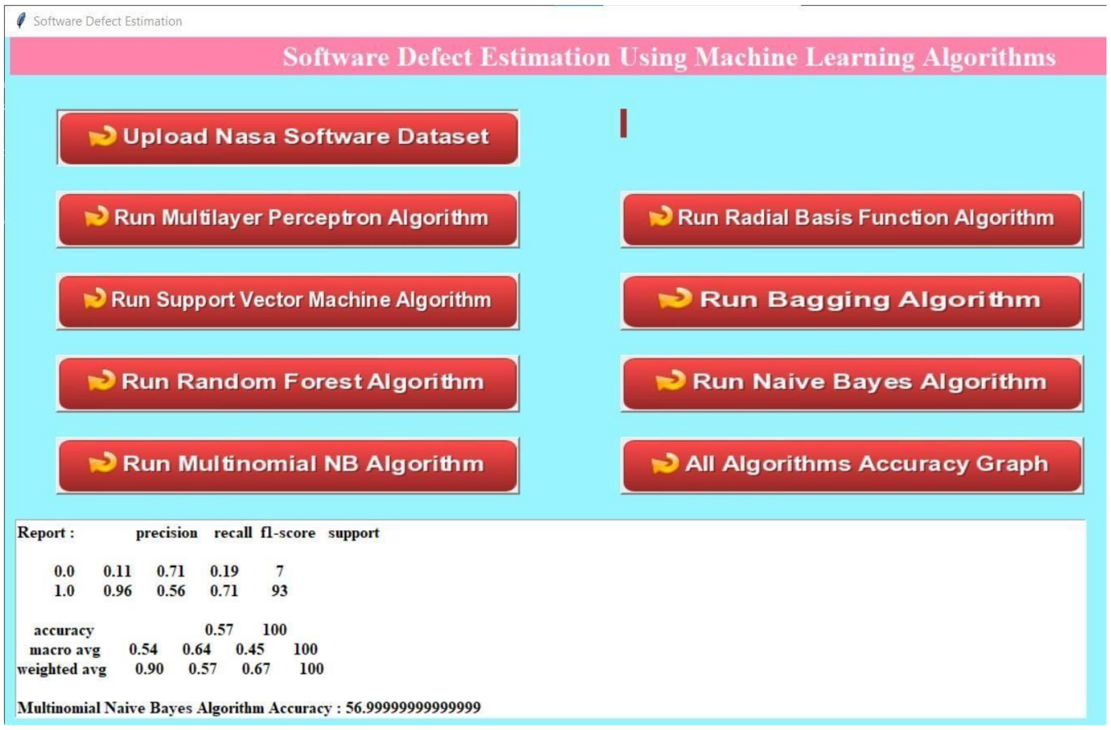
[ii] SVM or Support Vector Machine : It is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyper plane which separates the data into classes2


Software Defect Estimation
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :          precision    recall  f1-score   support

      0.0       0.00      0.00      0.00         7
      1.0       0.93      1.00      0.96        93

    accuracy                           0.93       100
   macro avg       0.47      0.50      0.48       100
weighted avg       0.86      0.93      0.90       100

Random Forest Accuracy : 93.0
```

Figure 14: Accuracy of RaF using CM1 dataset: 93.0

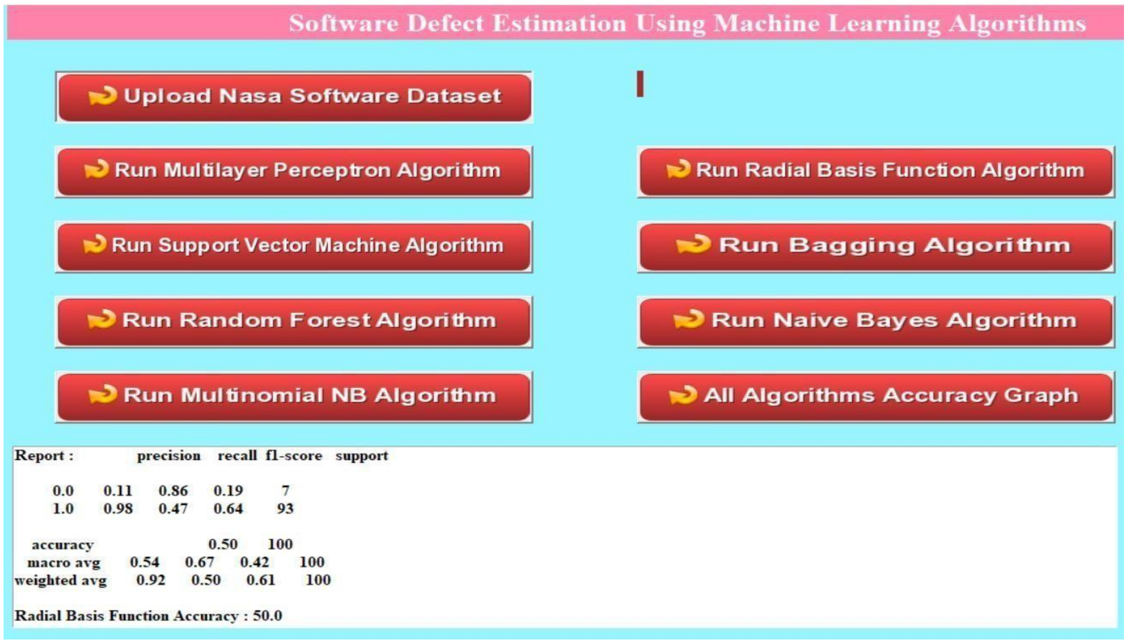
[iii] Random Forest Algorithm: it's an ensemble algorithm which means internally it will use multiple classifier algorithms to build accurate classifier model. Internally this algorithm will use decision tree algorithm to generate it train model for classification.


Software Defect Estimation
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :          precision    recall  f1-score   support

     0.0       0.11      0.71      0.19         7
     1.0       0.96      0.56      0.71        93

    accuracy                           0.57       100
   macro avg       0.54      0.64      0.45       100
weighted avg       0.90      0.57      0.67       100

Multinomial Naive Bayes Algorithm Accuracy : 56.99999999999999
```

Figure 15: Accuracy of Multinomial NB algorithm using CM1: 56.9999

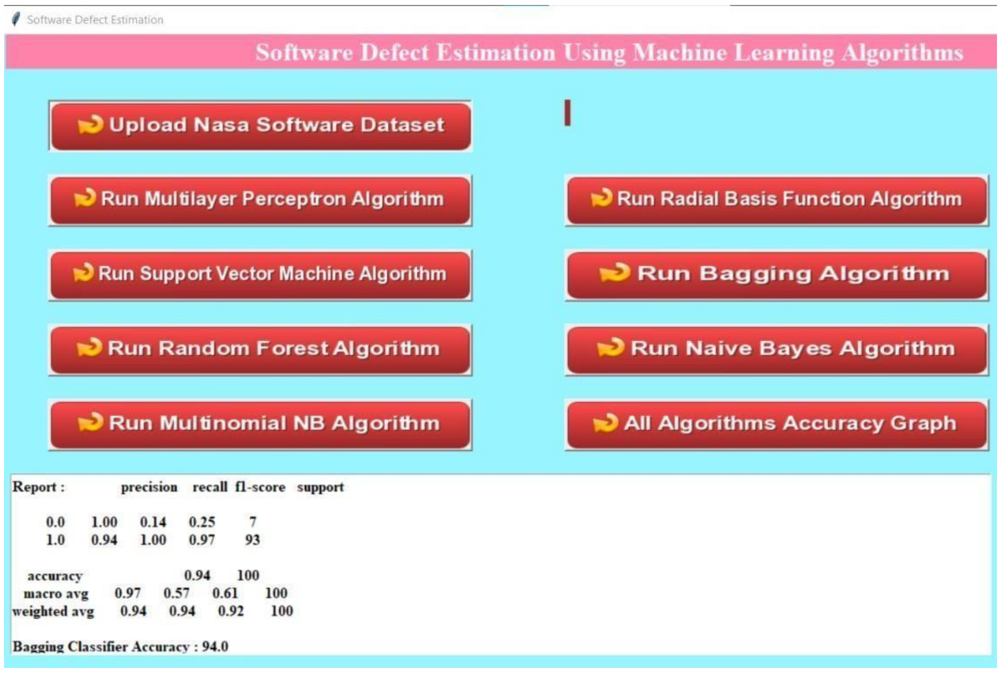
[iv] **Multinomial Naive Bayes:** Multinomial Naive Bayes classifier is obtained by enlarging Naive Bayes classifier. Differently from the Naive Bayes classifier, a multinomial distribution is used for each features.


Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :          precision    recall  f1-score   support

     0.0       0.11      0.86      0.19         7
     1.0       0.98      0.47      0.64        93

    accuracy                           0.50       100
   macro avg       0.54      0.67      0.42       100
weighted avg       0.92      0.50      0.61       100

Radial Basis Function Accuracy : 50.0
```

Figure 16: Accuracy of Radial Basis Function using CM1 dataset : 50.0

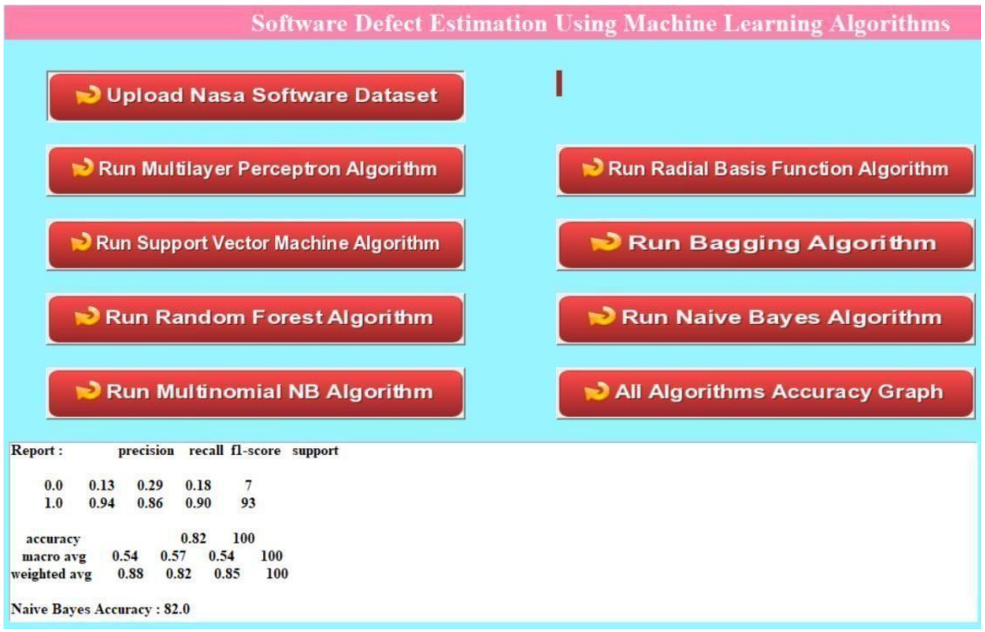
[v] **Radial Basis Function:** Network includes an input vector for classification, a layer of RBF neurons, and an output layer which has a node for each class. Dot products method is used between inputs and weights and for activation sigmoid activation functions are used in MLP while in RBFN between inputs and weights Euclidean distances method is used and as activation function, Gaussian activation functions are used.


Software Defect Estimation
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :         precision    recall  f1-score   support

     0.0       1.00      0.14      0.25         7
     1.0       0.94      1.00      0.97        93

    accuracy                           0.94       100
   macro avg       0.97      0.57      0.61       100
weighted avg       0.94      0.94      0.92       100

Bagging Classifier Accuracy : 94.0
```

Figure 17: Bagging Classifier using CM1 dataset : 94.0

[vi] **Bagging**: This algorithms work similar to learning tree the only difference is voting concept where each class will get majority of votes based on values close to it and that class will form a branch. If new values arrived then that new value will applied on entire tree to get close matching class.


Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :         precision    recall  f1-score   support

     0.0       0.13      0.29      0.18         7
     1.0       0.94      0.86      0.90        93

    accuracy                           0.82       100
   macro avg       0.54      0.57      0.54       100
weighted avg       0.88      0.82      0.85       100

Naive Bayes Accuracy : 82.0
```

Figure 18: Accuracy of Multinomial NB algorithm using CM1 dataset: 82.0

[vii] **Multinomial Naive Bayes:** Multinomial Naive Bayes classifier is obtained by enlarging Naive Bayes classifier. Differently from the Naive Bayes classifier, a multinomial distributionis used for each features.

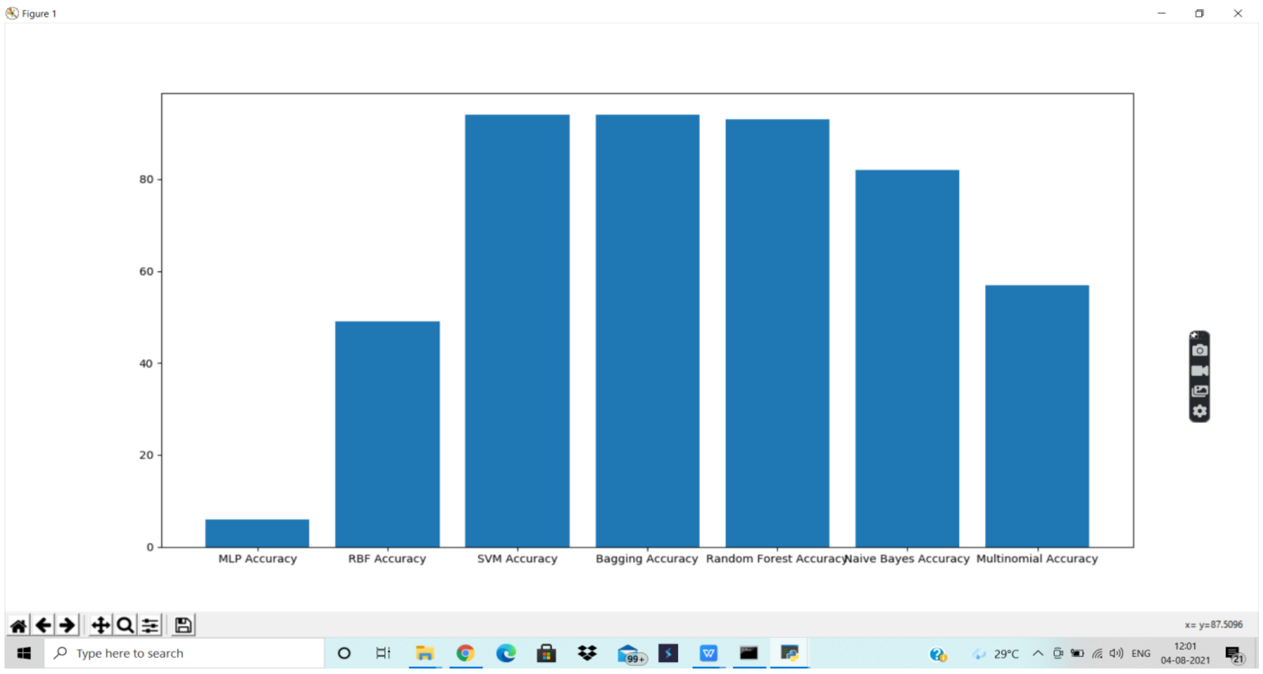
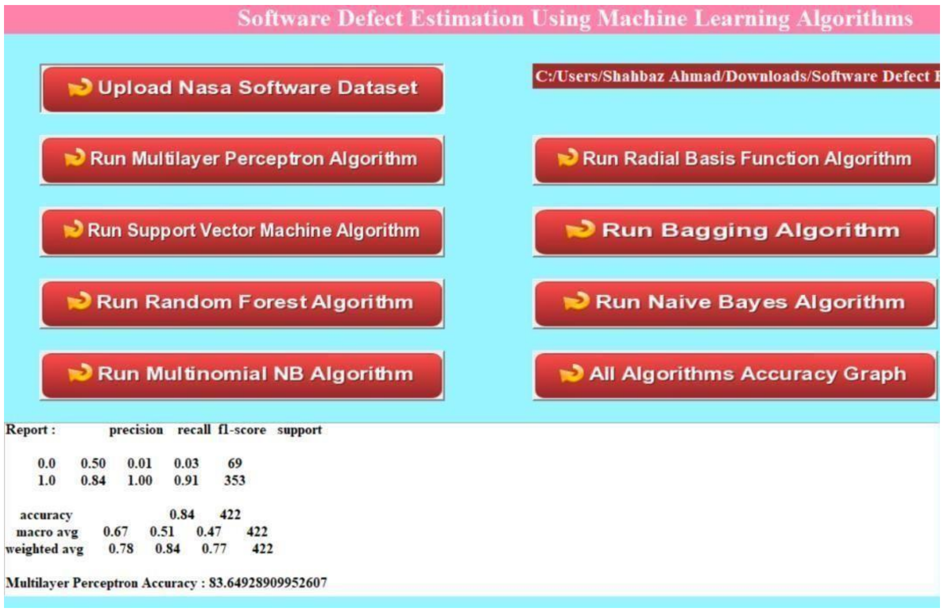


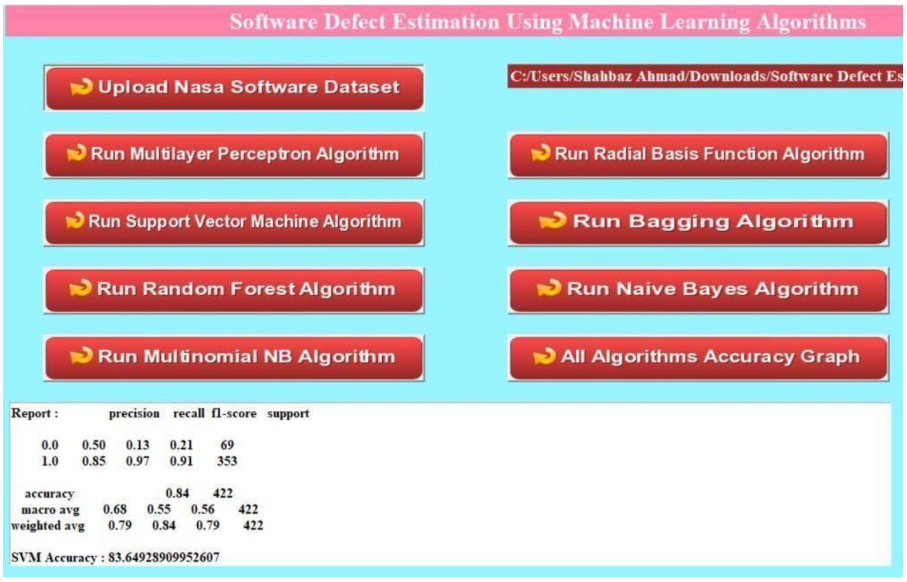
Figure 19: All Algorithms Accuracy Graph In CM1 dataset.

Here the above graph visualizing different accuracy for different algorithms mentioned in X - Axis whereas Y- Axis Represents different Accuracy Points shown in the above image.


Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
C:/Users/Shahbaz Ahmad/Downloads/Software Defect E
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

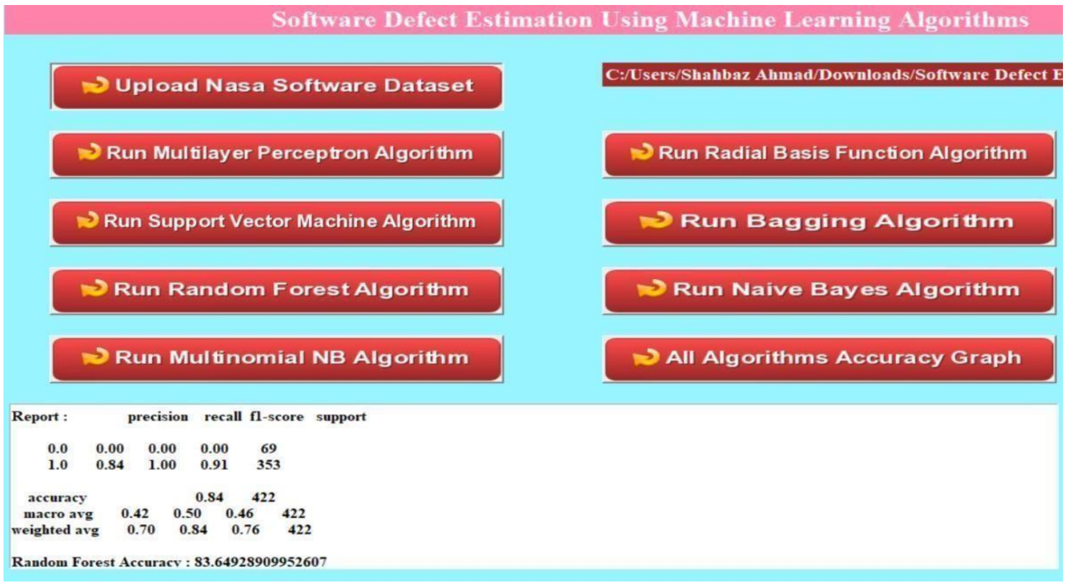
```
Report :          precision    recall  f1-score   support

     0.0       0.50      0.01      0.03        69
     1.0       0.84      1.00      0.91       353

  accuracy                          0.84       422
 macro avg       0.67      0.51      0.47       422
weighted avg     0.78      0.84      0.77       422

Multilayer Perceptron Accuracy : 83.64928909952607
```

Figure 20: Accuracy of Multilayer Perceptron algorithm using KC1: 83.6492


Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
C:/Users/Shahbaz Ahmad/Downloads/Software Defect Es
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

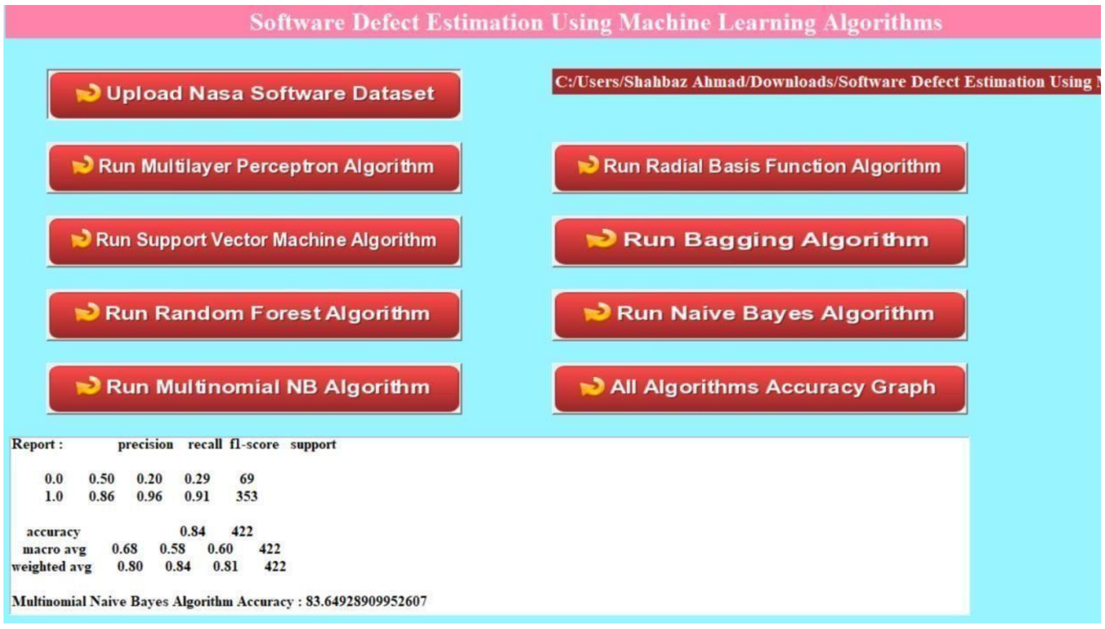
```
Report :          precision    recall  f1-score   support

     0.0       0.50      0.13      0.21        69
     1.0       0.85      0.97      0.91       353

  accuracy                          0.84       422
 macro avg       0.68      0.55      0.56       422
weighted avg     0.79      0.84      0.79       422

SVM Accuracy : 83.64928909952607
```

Figure 21: Accuracy of SVM using the KC1 dataset : 83.6492


Software Defect Estimation Using Machine Learning Algorithms
C:/Users/Shahbaz Ahmad/Downloads/Software Defect E
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

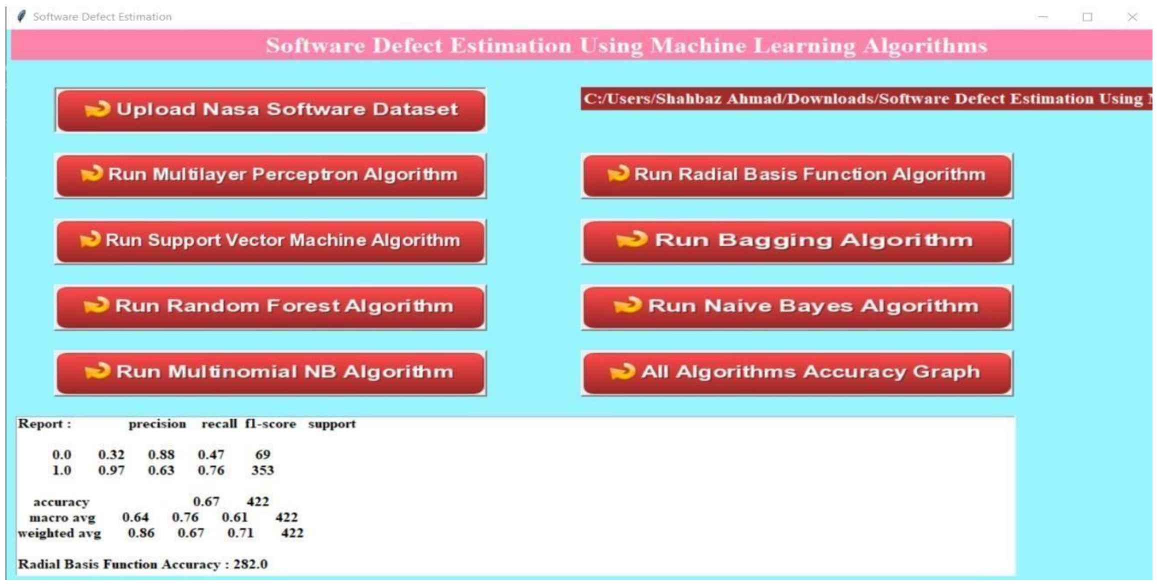
```
Report :           precision    recall  f1-score   support

     0.0       0.00      0.00      0.00        69
     1.0       0.84      1.00      0.91       353

    accuracy                           0.84       422
   macro avg       0.42      0.50      0.46       422
weighted avg       0.70      0.84      0.76       422

Random Forest Accuracy : 83.64928909952607
```

Figure 22: Accuracy of RaF using the KC1 dataset :83.6492


Software Defect Estimation Using Machine Learning Algorithms
C:/Users/Shahbaz Ahmad/Downloads/Software Defect Estimation Using
Upload Nasa Software Dataset
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

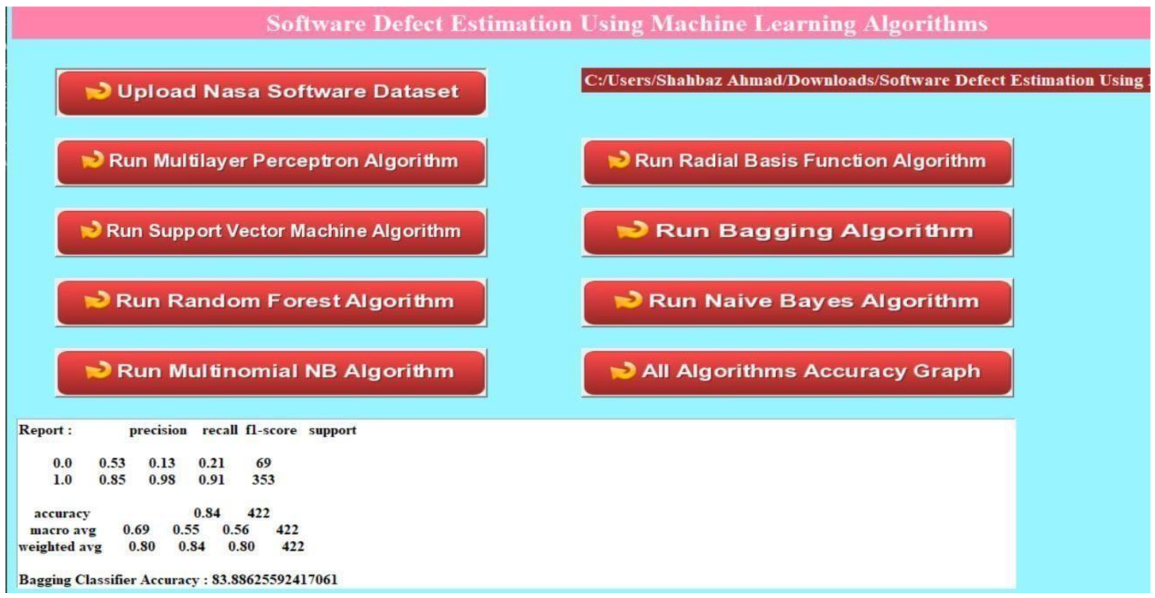
```
Report :           precision    recall  f1-score   support

     0.0       0.50      0.20      0.29        69
     1.0       0.86      0.96      0.91       353

    accuracy                           0.84       422
   macro avg       0.68      0.58      0.60       422
weighted avg       0.80      0.84      0.81       422

Multinomial Naive Bayes Algorithm Accuracy : 83.64928909952607
```

Figure 23: Accuracy of Multinomial NB algorithm using the KC1 : 83.6492


Software Defect Estimation
Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
C:/Users/Shahbaz Ahmad/Downloads/Software Defect Estimation Using
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

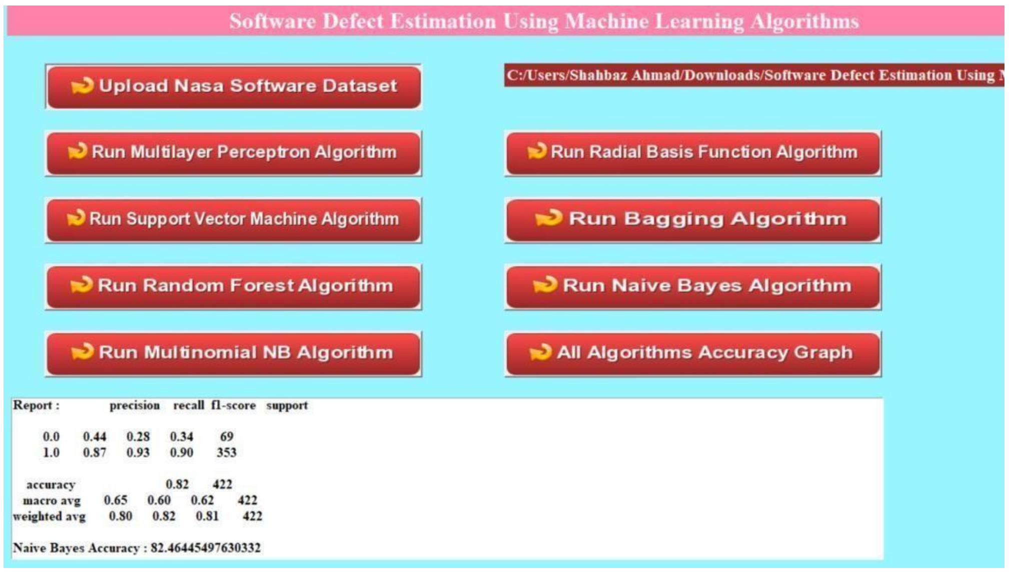
```
Report :          precision   recall  f1-score   support

     0.0       0.32      0.88      0.47        69
     1.0       0.97      0.63      0.76       353

  accuracy                           0.67       422
 macro avg       0.64      0.76      0.61       422
weighted avg     0.86      0.67      0.71       422

Radial Basis Function Accuracy : 282.0
```

Figure 24: Accuracy of Radial Basis Function algorithm using KC1 dataset:282.0


Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
C:/Users/Shahbaz Ahmad/Downloads/Software Defect Estimation Using
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :          precision   recall  f1-score   support

     0.0       0.53      0.13      0.21        69
     1.0       0.85      0.98      0.91       353

  accuracy                           0.84       422
 macro avg       0.69      0.55      0.56       422
weighted avg     0.80      0.84      0.80       422

Bagging Classifier Accuracy : 83.88625592417061
```

Figure 25: Accuracy of Radial Basis Function algorithm using KC1 dataset:83.8862


Software Defect Estimation Using Machine Learning Algorithms
Upload Nasa Software Dataset
C:/Users/Shahbaz Ahmad/Downloads/Software Defect Estimation Using
Run Multilayer Perceptron Algorithm
Run Radial Basis Function Algorithm
Run Support Vector Machine Algorithm
Run Bagging Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Multinomial NB Algorithm
All Algorithms Accuracy Graph

```
Report :          precision    recall  f1-score   support

      0.0       0.44      0.28      0.34        69
      1.0       0.87      0.93      0.90       353

  accuracy                          0.82       422
 macro avg       0.65      0.60      0.62       422
weighted avg     0.80      0.82      0.81       422

Naive Bayes Accuracy : 82.46445497630332
```

Figure 24: Accuracy of Naïve Bayes algorithm using KC1: 82.4644

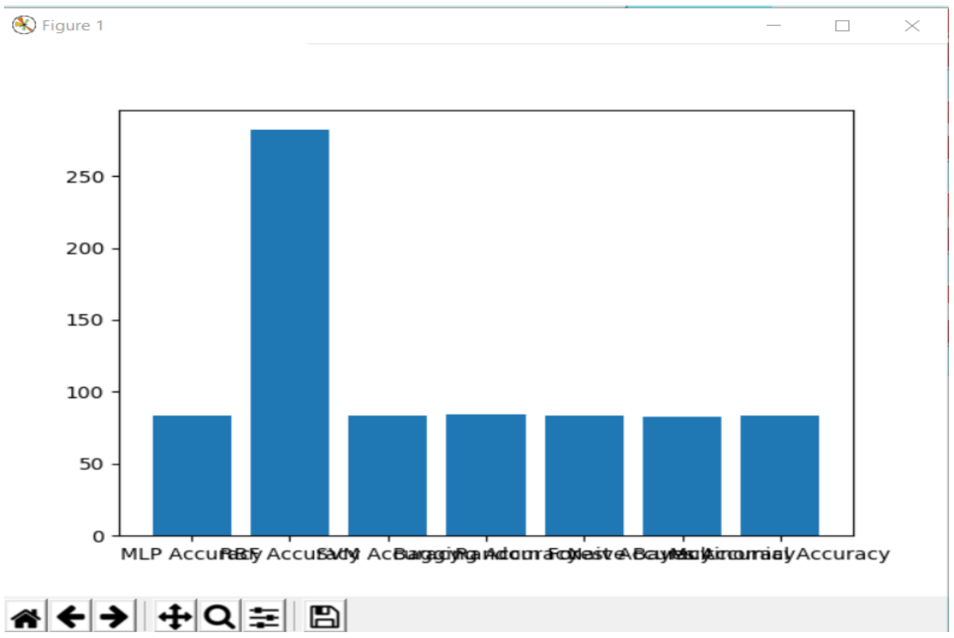


Figure 25: All Algorithms Accuracy Graph In KC1 dataset.

Here the above graph visualizing different accuracy for different algorithms mentioned in X - Axis whereas Y- Axis Represents different Accuracy Points usingKC1 dataset shown in the above image.

System Requirements

Hardware Requirement

- Processor: Pentium IVonwards
- Processor speed:24GHz
- RAM:4GB
- Hard disk space:40GB+
- Standard PC configuration to carryout challenging computing

Software Requirement

- Operating System: Windows 7 version & Higher
- Development Kit:JDK 1.7/JDK 1.8
- IDE: Jupiter Notebook

CONCLUSION

As the requirements increase, the IT industry has to do a lot of work to meet customer needs. As a result, data mining technology has been widely used to extract useful insights from large software repositories and accept it to detect bugs and quality of the software is improved. This project adapted the methods of detection and classification of software defects by incorporating machine learning techniques to identify and classify defectsfrom large software repositories. The activities described here use software defect metricsbased on the severity of defects in software products for classification. Random forestsget significantly better results than the other two classifications, KNN and Naive Bayes.

Future Enhancement

Further work on finding relational association rules is done to identify and account for various relationships types amongst software metrics and relationships that may be relevant to the extraction process. We will also analyze how length of the rule and reliability of the relational rules found in the training data influence the accuracy of the classification activities. By combining classification models with other predictive models based on machine learning, we can explore the direction to hybridize classification models.

REFERENCES

1) B. Cvetkoska, N. Marina, D. C. Bogatinoska and Z. Mitreski, Towards Effective Troubleshooting With Data Truncation, via Bug Repository Handling.Analysed by Additionally, text processing and data processing algorithm must be used to handle the bug repository.
2) HamidurRahman, Shankar Iyer, Caroline, MihaelaStoycheva, VukanTurkulov, Shahina Begum, and MobyenUddin Ahmed, ―A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage: Analysed by The order of applying instance selection and feature is selection must be determined by a prediction Algorithm.
3) Sara Colantonio, Giuseppe Coppini, Danila Germanize, Daniela Giorgi, "Automatic Bug Triage using Semi- Supervised Text Classification,‖ Biosystems Engineering", Techniques such as IS and FS must be used to reduce the data in bug repositorory Proper predictive algorithm must beused to classify bugs.
4) ChidambaramSethukkarasi, Vijayadharan, Raja Pitchiah, ―Approach of Data Reduction Techniques for Bug Triaging System on Adv in Intel Systems, the bugs in bug repository which can be minimized.
5) Nicolas Battenberg, Rahul Premraj, Thomas Zimmermann, and Asurveyon bug-report analysis . Several more techniques such as instance selection and feature selection must be used to handle the bugs.
6) Silvia Breu, Rahul Premraj, Jonathan Sillito, and Thomas Zimmermann. 2010. Information Needs in Bug Reports: IBug report network: varieties strategiesand impact inFos development. Tools must be introduced to handle BRN's complexity.