

Project Report (Major Project- II)

on

Bug Severity Prediction

Submitted in partial fulfillment of the requirements

for the award of the degree of

Master of Technology

in

Software Technology

By

Lokesh Chugh

Roll No.: - 2K16/SWT/504

Under the guidance of

Dr. Ruchika Malhotra

Associate Professor



Department of Computer Science & Engineering

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road, Delhi 110042

2019



Delhi Technological University
(Formerly Delhi College of Engineering)
Bawana Road, New Delhi-42

DECLARATION

I hereby declare that the thesis entitled “**Bug Severity Prediction**” which is being submitted to the Delhi Technological University, in partial fulfillment of the requirements for the award of the degree of Master of Technology in Software Technology is an authentic work carried out by me. The material contained in this thesis has not been submitted to any university or institution for the award of any degree.

DATE:

SIGNATURE:

LOKESH CHUGH

2K16/SWT/504

CERTIFICATE



Delhi Technological University
(Formerly Delhi College of Engineering)
Bawana Road, New Delhi-42

This is to certify that project report entitled “**Bug Severity Prediction**” done by me for the Major Project 2 for the award of degree of Master of Technology Degree in Software Technology in the Department of Computer Science & Engineering, Delhi Technological University, New Delhi is an authentic work carried out by me.

Signature:

Student Name

Lokesh Chugh

2K16/SWT/504

Above Statement given by Student is Correct.

Project Guide:

Dr. Ruchika Malhotra

Associate Professor

**Department of Computer Science
& Engineering**

**Delhi Technological University,
Delhi**

Acknowledgement

No volume of words is enough to express my gratitude towards my guide **Dr. Ruchika Malhotra**, Department of Computer Science & Engineering, Delhi Technological University, Delhi, who has been very concerned and has aided for all the materials essentials for the preparation of this project report. She has helped me to explore this vast topic in an organized manner and provided me all the ideas on how to work towards a research-oriented venture.

I am also thankful to **Dr. Rajni Jindal**, HoD of Computer Science & Engineering Department and **Dr. Ruchika Malhotra**, Coordinator, for the motivation and inspiration that triggered me for the project work.

I would also like to thank the staff members and my colleagues who were always there at the need of hour and provided with all the help and facilities, which I required, for the completion of my project work.

Most importantly, I would like to thank my parents and the almighty for showing me the right direction, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Lokesh Chugh
(2K16/SWT/504)

ABSTRACT

Assessment of the degree of deformities is basic so as to designate assets for testing and to plan inquire about exercises viably. We use content characterization methods in this paper to foresee and assess the seriousness of deformities. The outcomes depend on the deformity portrayal of Nasa venture issue necessities. We utilized the philosophy of Support Vector Machine to evaluate the recurrence of the issue pro-acclamations. In this examination study, a basic word lexicon approach is recommended to decide the earnestness of the bug as outrageous or non-genuine. It is discovered that the example of exactness and accuracy is generally a similar utilizing various methodologies of highlight choice and characterization. In any case, for each of the four parts, Chi square test and KNN classifier give most extreme exactness and precision effectiveness. The proposed work will assist Triage with identifying seriousness based bugs and designate these bugs to explicit engineers. The paper introduces another and robotized technique called Severity Problem Assessment that helps the test engineer in appointing levels of seriousness to reports of imperfections. Severity depends on conventional content mining and AI strategies for existing assortments of records of deformities. The paper gives a contextual analysis on the utilization of Severity and ITS(issue tracking system) of Nasa. The discoveries of the contextual investigation show that Severity is a decent indicator of the seriousness of the issue, while it is easy to utilize and powerful.

Table of Contents

CHAPTER 1	9
INTRODUCTION	9
1.1 EXISTING SYSTEM.....	10
1.2 PROPOSED SYSTEM.....	11
CHAPTER 2	13
RESEARCH METHODOLOGY	13
2.1 SEVERITY CLASSIFICATION PROCESS.....	13
2.2 DEFECT TRACKING SYSTEM.....	14
CHAPTER 3	18
SYSTEM DESIGN AND ARCHITECTURE	18
3.1 INTRODUCTION	18
3.2 SYSTEM DESIGN DOCUMENT	18
3.2.1 OVERVIEW	18
3.2.2 INTRODUCTION.....	18
3.2.3 SYSTEM ARCHITECTURE.....	18
3.2.4 FILE AND DATABASE DESIGN	19
3.2.5 HUMAN-MACHINE INTERFACE	19
3.2.6 DETAILED SYSTEM ANALYSIS:	19
3.2.7 SYSTEM INTEGRITY CONTROLS.....	20
3.2.8 THE SEVERIS:.....	20
CHAPTER 4	24
KNN ALGORITHM.....	24
4.1 K-NEAREST NEIGHBORS' CLASSIFICATION	24
4.2 PERFORMANCE ANALYSIS.....	25
4.2.1 PERFORMANCE PARAMETERS	25
4.2.2 COMPARISON PARAMETER:.....	26
4.3 SYSTEM TESTING.....	27
CHAPTER 5	28
THE SOFTWARE PLATFORM.....	28
5.1 JAVA.....	28

5.2 JAVA PLATFORM.....	28
5.3 TOOLS USED	29
5.4 IMPLEMENTATION:	30
5.4.1 THE ADMIN MODULE.....	30
5.4.2 THE USER MODULE:.....	35
CHAPTER 6.....	38
RESULTS	38
CHAPTER 7.....	40
CONCLUSION AND FUTURE WORK	40
FUTURE WORK:	40
REFERENCES:.....	41

List of Figures

Figure 2.1: Detailed Methodology.....	14
Figure 2.2 : Top 25 terms in dataset after Feature Selection	16
Figure 2.3: Training for 5 Features.....	17
Figure 2.4: Training for 50 Features.....	17
Figure 3.1 : Workflow of Severis	22
Figure 3.2 : System Architecture of Bug Detection.....	23
Figure 4.1 : Distance Functions	24
Figure 4.2 : Confusion Matrix	27
Figure 5.1 : Java Program Flowchart.....	29
Figure 5.2 : Java Platform.....	29
Figure 5.3 : Sample Dataset.....	31
Figure 5.4 : Admin Login	32
Figure 5.5 : Upload Dataset	32
Figure 5.6 : View Dataset	33
Figure 5.7 : Pre-Processing Data	33
Figure 5.8 : Feature Extraction	34
Figure 5.9 : Calculate TF-IDF Score	34
Figure 5.10 : View Bug Severity	34
Figure 5.11 : User Registration.....	35
Figure 5.12 : User Login.....	35
Figure 5.13 : View Profile	36
Figure 5.14 : Add Bugs.....	37
Figure 5.15 : View Severity Level.....	37
Figure 6.1 : Accuracy of trained model	38
Figure 6.2 : Prediction Results.....	39

CHAPTER 1

INTRODUCTION

With the expanding reliance on programming frameworks, significance of programming quality is getting increasingly basic. There are various approaches to guarantee quality in programming, for example, code surveys and thorough testing with the goal that bugs can be evacuated as right on time as conceivable to avoid the misfortune it might cause. Distortion reality assessment is generally stressed over evaluation of significant worth degree of an item to see whether the item is fit enough to be released. In every way that really matters, the flaw reports are filled physically on intermittent reason. The estimation of reality levels doled out to gives up physically could be not exactly equivalent to enrolled by content mining and rule learning. To forestall a engineer from allotting incorrectly seriousness levels to an imperfection, to spare time if there should be an occurrence of an earnest time limited application, to watch that no high seriousness issue has been missed, a mechanized extraction and examination of content from issue reports has been utilized [1][2]. Defect severity assessment is principally involved with assessment of quality level of a package to visualize whether or not the package is work enough to be free. much, the defect reports are stuffed manually on periodic basis. the worth of severity levels appointed to defects manually is totally different from text mining and rule learning process.

In this paper we've a bent to gift a replacement approach for extracting general conclusions from PITS info supported text mining and machine learning ways in which, that low value, automatic, and rapid. we've a bent to designed and designed a tool named SEVERIS automatically review issue reports and alert once a projected severity is abnormal. The severis is created provides the probabilities that the assessment is correct. These possibilities are going to be used to guide method} throughout this method. distribution the correct severity levels to issue reports is extraordinarily necessary within the strategy used at office, as a result of it directly impacts resource allocation and developing with of later defect fixing activities [8].

A. Severis Content mining methods are utilized to separate the applicable highlights of each report, while AI procedures are utilized to relegate these highlights with appropriate seriousness levels, in light of the characterizations of existing reports [8].

B. Bug: A Software bug can be delegated mistake, blemish, disappointment or flaw in any framework because of which framework act in an ill-advised way, may give results which are not expected or wrong outcomes. Different ways by which a bug can occur because of blemishes in source code, planning of program or because of working frameworks or additionally can be review of bugs assessment of bugs. The after effects of bugs finished up to be unsafe, from different episodes in genuine world [3].

1.1 EXISTING SYSTEM

There are various approaches to guarantee quality in programming, for example, code surveys and thorough testing with the goal that bugs can be expelled as ahead of schedule as conceivable to avoid the misfortune it might cause. Programming bug is generally used to portray the event of a shortcoming in a product framework which results it to act uniquely in contrast to its determination bug following frameworks are one of the significant stores among all accessible programming storehouses. Many open source programming tool have an open bug vault that permits the two engineers and clients to submit imperfections or issues in the product, propose potential improve and remark on existing bug reports.

Disadvantages:

- Another issue basic to these frameworks is that the majority of the information is unstructured. Explicit to [PITS] is that the database fields in [PITS] continue changing,yet the idea of the unstructured content stays steady [5].
- Many open source programming ventures have an open bug archive that enables the two designers and clients to submit imperfections or issues in the product, recommend potential upgrades, and remark on existing bug reports.
- As contrasted with more up to date frameworks, the issue with [PITS] is that there is an absence of consistency in how every one of the undertakings gathered issue information.

1.2 PROPOSED SYSTEM

As the report is first sent into the information, it is perused and sent for being pre-prepared which includes the decrease of trait so the superfluous and pointless words present in the huge measure of information could be restricted to just the characteristics that are essentially significant. the pre-handled words are sent for include determination in order to keep up a degree of consistency among the traits by positioning them from generally essential to fundamentally less significant. The traits are then applied with AI calculation in order to characterize the qualities based on seriousness levels of their imperfections. The outcomes got from AI are then assessed utilizing f-measure, which incorporates exactness and review, to know how much the outcomes acquired hold fast to the ideal yield.

Reducing the number of attributes so as to get only the relevant and significantly important attributes in order to save time in processing the entire chunk of data and shift our focus on relevant attributes. Data pre-processing encompasses three steps in order as: tokenization, stop word removal, and stemming.

Advantage:

- A. Tokenization Process:** It is the assignment of changing over a flood of characters into a surge of handling units called tokens, which are the squares of content thought about helpful piece of the unstructured content [6].
- B. Stop Word Removal:** Stop words are the most well-known words that are not prone to convey any huge data in a specific setting and are, in this way, a bit much for content mining application and are wiped out [6].
- C. Stemming:** Stemming is the procedure for lessening bent or now and then determined words to their stem base or root structure commonly a composed word structure. For instance: run, runs, ran, and running, are largely types of a similar root, customarily composed as run and the job of a stemmer is to characteristic all the determined structures to the base of the lexeme [6].
- D. Feature Selection:** Much after information pre-preparing on a record, the quantity of characteristics is huge to the point that it further should be decreased. Highlight choice

exemplifies the method for finding the significance of a word based on different strategies, for example, Info Gain [6].

E. Term frequency (TF) and Inverse Document Frequency (Idf):

Tf and idf is shorthand for term recurrence times reverse archive recurrence. The tf and idf weight characterized for a word is frequently utilized in data recovery and content mining. This weight is a factual measure used to assess how significant a word is to a record in an assortment or corpus. The significance expands relatively to the occasions a word shows up in the report however is counterbalanced by the recurrence of the word in the corpus. The term recurrence in the given archive is basically the occasions a given term shows up in that record.

CHAPTER 2

RESEARCH METHODOLOGY

2.1 SEVERITY CLASSIFICATION PROCESS

Bug reports are removed from separate bug archive. At that point pre-handling on printed data of bug reports is applied to acquire progressively solid data. Term-record grid is made and by utilizing highlight determination strategy lexicon of basic terms is made. At that point decreased TDM got by utilizing basic word reference terms is sustained to classifier for characterization of serious and non-extreme bug report. The entire procedure of seriousness characterization process proposed in the theory can be outlined as underneath.

Figure 2.1 shows the following steps:

Step 1: Extraction of defective bug reports of open source software from bug repository.

Step 2: Preprocessing occurs when defective bug reports by using text mining approach

Step 3: Use TF / IDF score for creation of a term document matrix.

Step 4: The use of a feature selection method selection methods, information gain and Chisquare method for dimensionality reduction.

Step 5: Creation of critical term dictionary using top-k terms that are obtained after dimensionality reduction and will be fed to the hybrid algorithm proposed in the thesis.

Step 6: The establishment of Hybrid KNN-NBM algorithm for improved Bug severity prediction.

Step 7: Severity process dection of the bugs reported.

Step 8: The dataset that we are using will be used from Bugzilla repository of mozilla firebox OS.

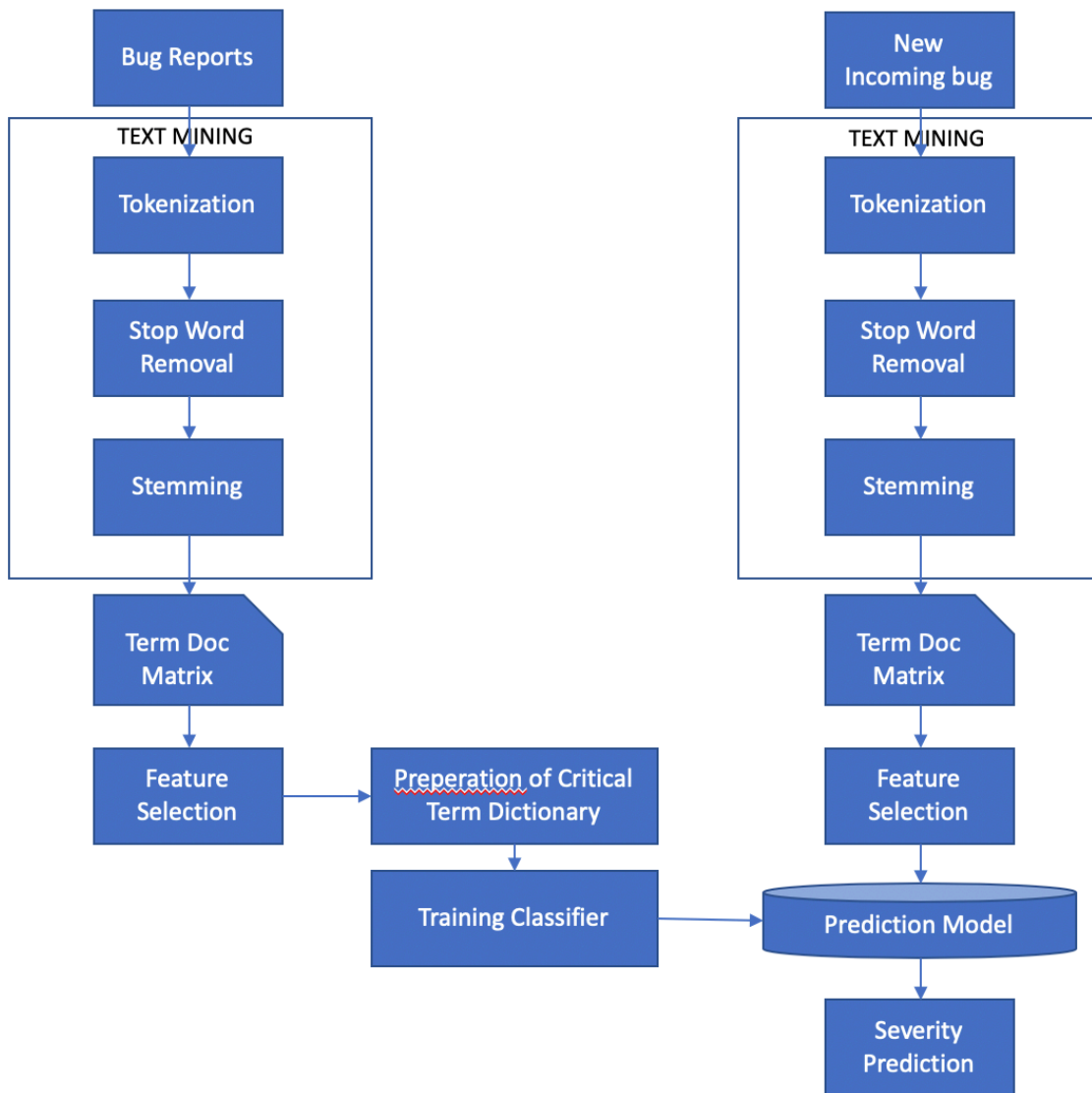


Figure 2.1: Detailed Methodology

2.2 DEFECT TRACKING SYSTEM

It is a fundamental advance for venture the board. A deformity report shows the manner in which a client sees the imperfection. For the duration of the existence cycle of the imperfection report, it experiences a work process which comprises of six phases as new, Examined, Confirmed, Resolved, Verified and Accepted [7].

At the point when an imperfection report is submitted into the framework, it is first arranged as New. It is then guaranteed for quality administration by the quality affirmation faculty, at that point it is sent into the analyzed state. Presently, when the deformity report is seen as causing a disappointment in the activity of the part or a framework, it is sent to the affirmed state. On the off chance that an answer is resolved to address the deformity experienced, the imperfection report is moved into the settled status. The quality confirmation faculty at that point check the answer for right the imperfection and if the redress is done accurately, the deformity report currently moves to the confirmed state. Presently the arrangement is displayed to the customer to be endorsed; on the off chance that it is affirmed, the deformity is moved to the acknowledged state.

There can be a situation when a deformity report contains approaches to address the imperfection, it can legitimately be moved from new or inspected to settled. Significantly after an imperfection is in affirmed state and the data present has been seen as off base, it can follow back to inspected state for further handling from affirmed. In the event that the remedy of the deformity isn't right, it is revived from settled state to inspected [9].

The PITS A dataset was utilized for turning out with the outcome. 70% of the archives were utilized for preparing and the rest 30% were utilized for testing the exactness and productivity the preparation strategy and in this manner anticipating the viability of the content arrangement.

Figure 2.2 shows list of top 25 terms in the data set after feature selection has been done.

Figure 2.3 and Figure 2.4 shows graphs of Training Features with 5 and 50 features respectively.

Rank	Feature
1	rvm
2	question
3	lead
4	script
5	system
6	3
7	test
8	red
9	car
10	file
11	dh
12	e
13	miss
14	line
15	engine
16	reject
17	delink
18	scrub
19	av
20	tech
21	fiber
22	group
23	center
24	motor
25	radio

Figure 2.2 : Top 25 terms in dataset after Feature Selection

The Graph of Feature Selection:

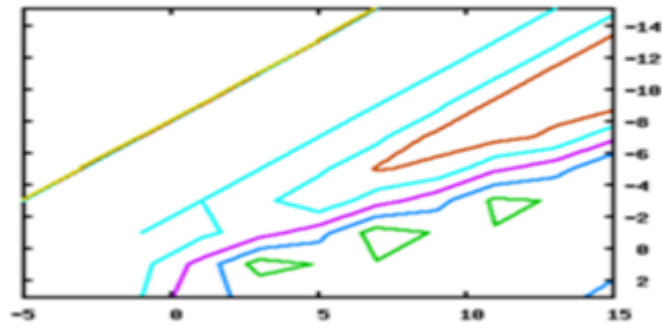


Figure 2.3: Training for 5 Features

Best $\log_2(C) = 7$ $\log_2(\gamma) = -3$ accuracy = 75.28%
 $C = 128$. $\gamma = 0.125$

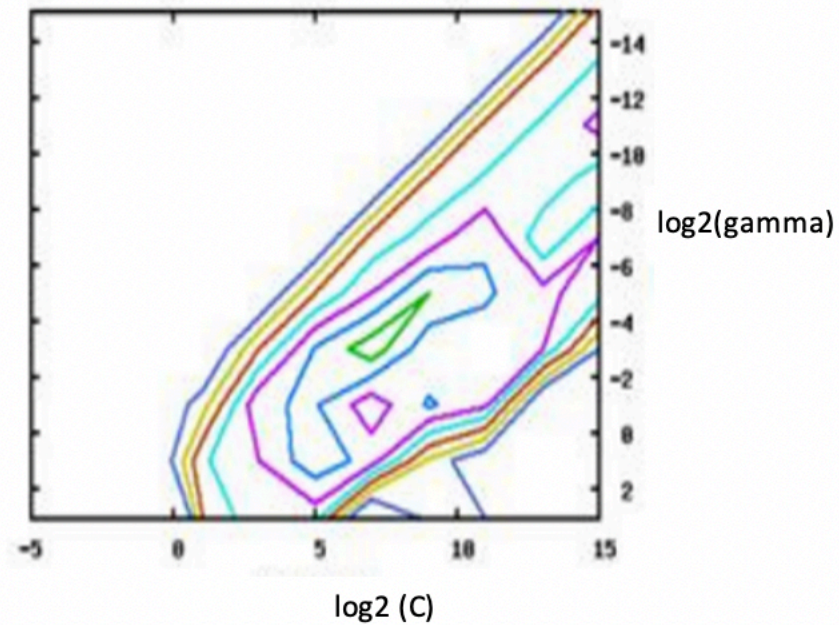


Figure2.4: Training for 50 Features

When selected total 100 features, we got an accuracy of 84%. Figure 4 is a plot of the training data based on SVM using LIBSVM.

CHAPTER 3

SYSTEM DESIGN AND ARCHITECTURE

3.1 INTRODUCTION

The product bugs that are recognized after the sending of programming influence the dependability and nature of programming. Bug following frameworks enables clients to report these bugs of many open source programming. Be that as it may, anticipating the seriousness level of these bug report is raising issue [10].

3.2 SYSTEM DESIGN DOCUMENT

3.2.1 OVERVIEW

This document shows various requirements of the system, the operating system, files, database, input, output, various interfaces, logic implemented, internal interface and external interface.

3.2.2 INTRODUCTION

Introduction part describes the following things:

1. Purpose and scope of the implementation
2. The summary of the project to be implemented from management view
3. The overview of the system using technical and non-technical. The flowchart and layout of the project with appropriate diagrams.
4. Various constraints that are seen while implementing the project and any imaginations/exceptions made by the team while developing.
5. Future problems that may arise after developing the system
6. Various stakeholders involved while implementing. These can be the Project Manager, QA Manager, Security, Configuration, Organization, etc.
7. Various references that have been used while implementing the project
8. A summary of abbreviations and short forms that have been used.

3.2.3 SYSTEM ARCHITECTURE

This describes the architecture of the system that has been used. The architecture can be of the hardware or the software level. Various architectures that are described are as follows:

1. System Hardware Architecture which includes the complete architecture of system including various components.
2. System Software Architecture which described the software part, language, functions, tools, classes, Object-oriented diagrams, etc.
3. Internal Communication Architecture which describes the communication between the various components and modules of the system.

3.2.4 FILE AND DATABASE DESIGN

Along with the hardware and software part, the next important component is the file and database design. This includes the interaction of the Database Administrator with the various files (Both DBMS and non-DBMS files) which are related to the development of the project. It also describes how the data is stored in the DBMS and what are the various schemas, sub-schemas, tables, records, sets, etc used while storing the data in the DBMS. Various methods to access the tables and records and the size of the database and tables described [11][12].

Non- DBMS files includes the description of the files with proper input and output. This also includes methods of how to access the files in the database, that is, the keys and indexes or any other reference data that has been used, ways to access the files that have been stored in the DBMS.

3.2.5 HUMAN-MACHINE INTERFACE

Along with the hardware, software and communication with the database, the hardware must interact with humans as well. This section explains the interaction of the hardware component with humans. From the initial start of the project, the person using the project should be able to understand the flow of the program and the next step involved in the program. The interaction between machine and human should be smooth enough for the proper flow. Various parts involved in this interface are Inputs and Outputs [13].

3.2.6 DETAILED SYSTEM ANALYSIS:

This describes the designing of all the components involved which are hardware, software, interaction between various modules, etc. Depending upon the implementation all the details are required, some are as described below:

- Details of various hardware components used
- Various connectors / cables required
- Power requirement for input
- Memory requirement / Storage Space requirement
- Processor Speed and functionality
- Switches and cables used
- GUI of all the components used in the hardware
- Functions , algorithms and interaction of various modules used in the implementation
- Data Entry Methods and various ways to access the records and files structures.
- Various elements used in saving the data to a particular storage area.
- Communication ways used to transfer the data
- Topology of the cables and connections
- Number of Clients and server used in maintaining the connection

3.2.7 SYSTEM INTEGRITY CONTROLS

This includes ways to recover the data in case of complete failure, unauthorized access or misuse of the implemented system. The system must be accessible and must be recoverable in such cases and there should be proper security measures to recover the system in all such cases. There should be proper review and audit system that must occur on a particular stage of the implementation. Security must be of concern so that the implementation is not openly available to all and must be controlled by limiting the number of users. All the data entered must be verified with proper sources [14].

3.2.8 THE SEVERIS:

Severis depends on the computerized extraction and investigation of literary depictions from issue reports in PITS. Content mining procedures are utilized to remove the significant highlights of each report, while AI strategies are utilized to allot these highlights with appropriate

seriousness levels, in light of the groupings of existing reports. While, in its present structure is explicitly custom-made to work with PITS reports, with little alterations, Severis can be utilized with other deformity detailing frameworks, for example, bug in programs [8].

The delineates how Severis entomb works with the human examiner or his manager. Severis checks the legitimacy of the seriousness levels relegated to issues in the accompanying manner: After observing an issue in some antiquity, a human investigator produces some content notes and appoints a seriousness level seriousness. Severis learns an indicator for issue seriousness level from logs of notes, seriousness.

Severis checks the severity level as mentioned below :

1. The updating of the Severis beliefs and
2. It Shows how much self confidence a manager might have in the severis conclusions.

The learned knowledge, Severis reviews the analyst's text and generates its own severity level.

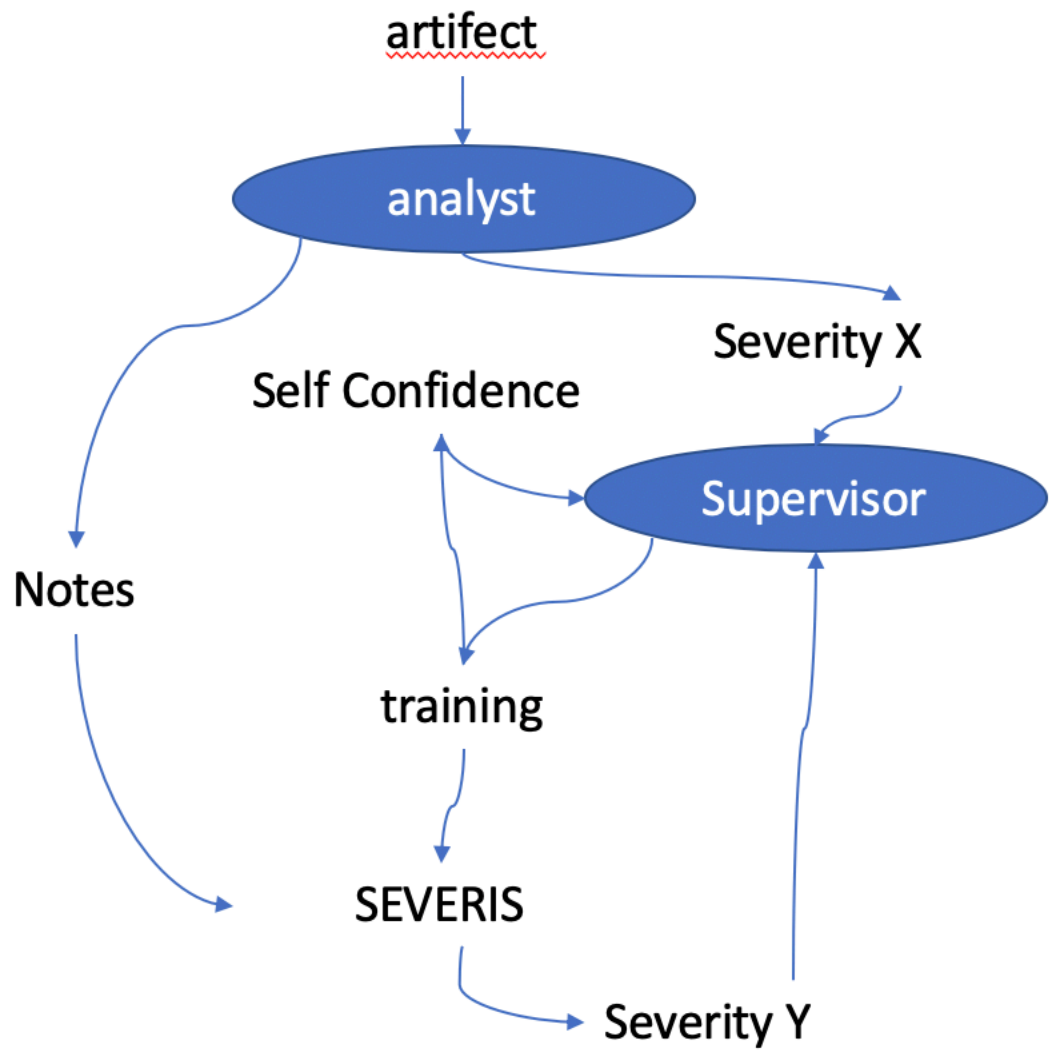


Figure 3.1 : Workflow of Severis

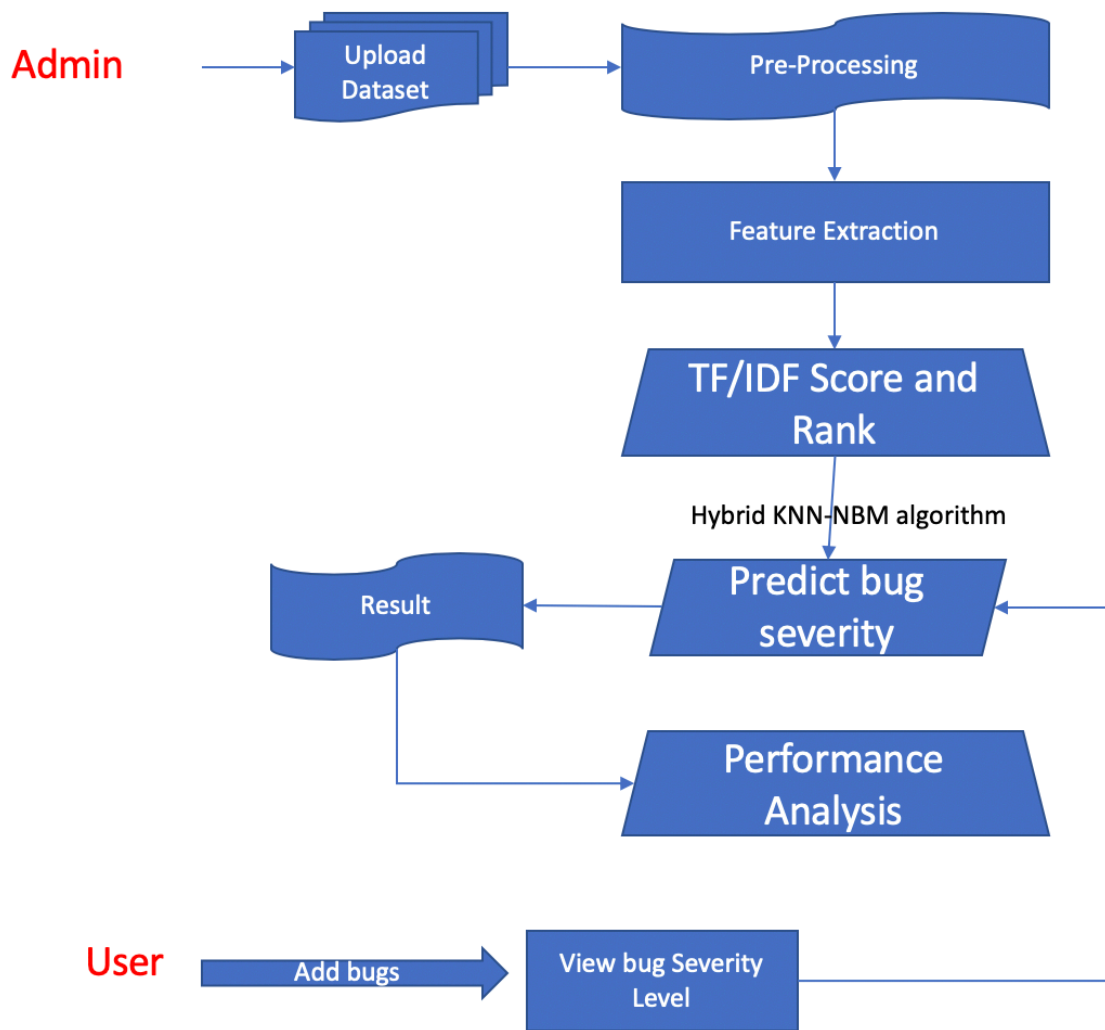


Figure 3.2 : System Architecture of Bug Detection

CHAPTER 4

KNN ALGORITHM

4.1 K-NEAREST NEIGHBORS' CLASSIFICATION

K closest neighbors is a straightforward calculation that stores every single accessible case and characterizes new cases dependent on a closeness measure (e.g., separation capacities). KNN has been utilized in measurable estimation and example acknowledgment as of now in the start of 1970's as a non-parametric strategy [15].

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function(Figure 4.1) , If $K = 1$, then the case is simply assigned to the class of its nearest neighbor as shown

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2} \quad \text{Euclidiean}$$
$$\sum_{i=1}^k |x_i - y_i| \quad \text{Manhattan}$$
$$\left(\sum_{i=1}^k (|x_i - y_i|^q) \right)^{1/q} \quad \text{Minkowski}$$

Figure 4.1 : Distance Functions

4.2 PERFORMANCE ANALYSIS

4.2.1 PERFORMANCE PARAMETERS

These are the exhibition parameters on which our calculation exactness, effectiveness and multifaceted nature would be estimated.

Accuracy

Accuracy is the rate of proportion of the total number of predictions that are correct in the system

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

Precision

Exactness or Confidence (as it is brought in Data Mining) indicates the extent of Predicted positive cases that are effectively real positives.

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

Recall

Review or Sensitivity as it is brought in Psychology is the extent of Real Positive cases that are effectively Predicted Positive.

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

4.2.2 COMPARISON PARAMETER:

These parameters help us in comparing our algorithm with other algorithms and techniques which are used for software quality prediction.

1. Confusion Matrix: A table of confusion (sometimes also called a confusion matrix). Bug Reports term document matrix new incoming bug term document matrix feature selection preparation of critical term dictionary feature selection training classifier prediction model severity prediction text mining stop word removal stemming text.
2. Mining tokenization Stemming Stop word expulsion is a table with two lines and two sections that reports the quantity of bogus positives, bogus negatives, genuine positives, and genuine negatives.

At the point when we get the, a great many pieces of information cleaning, pre-handling and wrangling, the initial step we do is to sustain it to an exceptional model and obviously, get yield in probabilities. In any case, hang on! How in the damnation would we be able to quantify the viability of our model? Better the adequacy, better the presentation and that is actually what we need. Also, it is the place the Confusion network comes into the spotlight. Disarray Matrix is a presentation estimation as demonstrated as follows.

		Predicted Condition	
		Predicted Condition(P ositive)	Predicted Condition(N egative)
Actual Condition	Total Population		
	Actual Condition(P ositive)	A: True Positive	B: False Negative
	Actual Condition(N egative)	C: False Positive	D: True Negative

Figure 4.2 : Confusion Matrix

4.3 SYSTEM TESTING

The reason for testing is to find mistakes. Testing is the way toward attempting to find each possible deficiency or shortcoming in a work item. It gives an approach to check the usefulness of segments, sub-congregations, gatherings or potentially a completed item.

TYPES OF TESTS

Unit testing:

Unit testing includes the structure of experiments that approve that the interior program rationale is working appropriately, and that program inputs produce legitimate yields.

Integration testing:

Reconciliation tests are intended to test incorporated programming parts to decide whether they really run as one program.

Functional test:

Utilitarian tests give methodical exhibitions that capacities tried are accessible as determined by the business and specialized necessities, framework documentation, and client manuals.

White Box Testing:

White Box Testing is a trying wherein in which the product analyzer knows about the internal functions, structure and language of the product, or if nothing else its motivation.

CHAPTER 5

THE SOFTWARE PLATFORM

5.1 JAVA

Java is both a high-level language and a platform independent. For implementation of this project Java language is used since Java is Simple, It's a Object oriented , Portable, High performance, Dynamic, Integrated and Secure. Figure 5.1 explains the basic flow of a Java Program.

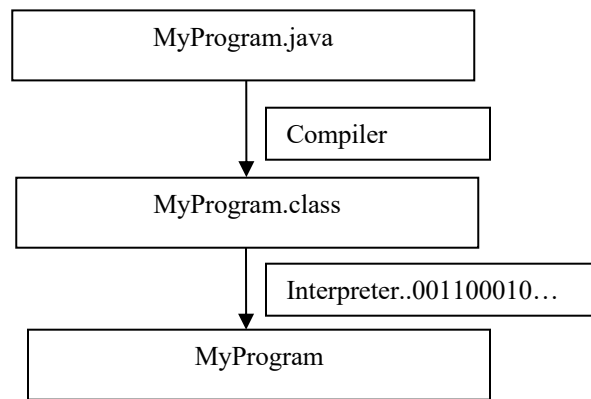


Figure 5.1 : Java Program Flowchart

5.2 JAVA PLATFORM

Java software has 2 parts as described below(Figure 5.2):

- The Java Virtual Machine (JVM) .
- The Java Application Programming Interface (Java API).

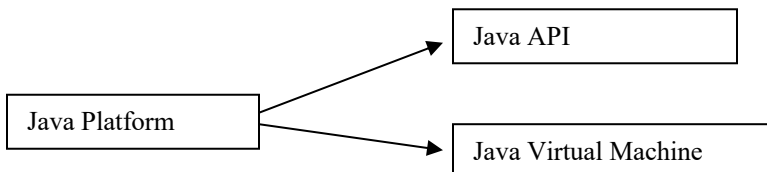


Figure 5.2 : Java Platform

5.3 TOOLS USED

OPERATING SYSTEM:

Windows 10

The project was performed on a Microsoft Windows 10 machine with 4GB RAM, intel Core i3 processor, 32MB cache memory. Code implementation was done with Java Language, apache-tomcat-6.0.20, eclipse-jee-mars-2-win32-x86_64, SQLSERVER used for creating web applications.

DATABASE:

MYSQL

- MySQL is an open-source, fast reliable, and flexible relational database management system, typically used with PHP. This chapter is an introductory chapter about MySQL, what is MySQL, and the main features of MySQL are described
- Installing MySQL on one window is relatively simple. You only need to download the MySQL installation package for the window version and install the installation package.
- A Sample Dataset image(Figure 5.3):

Bug ID	Product	Component	Assignee	Status	Resolution	Summary	Changed	Assignee	Number of Opened	OS	Priority	Reporter	Severity	Version	
3638	JDT	UI	aeschli	VERIFIED	FIXED	Package V	1/17/2002 7:28	Martin Ae	3	10/10/2001 22:58	Windows	P1	aeschli	major	2
3854	JDT	UI	aeschli	VERIFIED	FIXED	Wrong ex	1/18/2002 4:02	Martin Ae	5	10/10/2001 23:01	Windows	P1	david_auc	normal	2
4188	JDT	UI	aeschli	VERIFIED	FIXED	type hiera	1/28/2002 3:12	Martin Ae	3	10/10/2001 23:07	Windows	P1	erich_garr	normal	2
5115	JDT	Debug	aeschli	VERIFIED	FIXED	Workspac	11/13/2001 10:11	Martin Ae	10	10/19/2001 13:41	Windows	P1	darin.eclij	normal	2
5820	JDT	UI	aeschli	VERIFIED	FIXED	Close all e	11/20/2001 16:22	Martin Ae	5	11/12/2001 18:18	Windows	P1	jed.ander	normal	2
6441	JDT	UI	aeschli	VERIFIED	FIXED	"Plugins c	1/16/2002 2:58	Martin Ae	2	11/29/2001 13:50	Windows	P1	erich_garr	normal	2
7329	JDT	UI	aeschli	VERIFIED	FIXED	Outline vi	1/15/2002 5:56	Martin Ae	2	1/8/2002 2:44	Windows	P1	daniel_m	normal	2
7966	JDT	UI	aeschli	VERIFIED	FIXED	Class Crea	3/28/2002 5:43	Martin Ae	4	1/21/2002 15:42	Windows	P1	Tod_Creas	normal	2
188521	Platform	User Assis	agarcher	VERIFIED	FIXED	[Webapp]	6/11/2007 9:50	Adam Arc	15	5/22/2007 20:12	All	P1	mpawlow	normal	3.3
6341	JDT	UI	akiezun	VERIFIED	FIXED	NullPointerException	1/16/2002 2:31	Adam Kie	2	11/27/2001 5:58	Windows	P1	aeschli	normal	2
7046	JDT	UI	akiezun	VERIFIED	FIXED	NPE trying	1/15/2002 6:14	Adam Kie	2	12/18/2001 10:49	other	P1	eclipse	normal	2
7317	JDT	UI	akiezun	VERIFIED	FIXED	ClassCastE	1/15/2002 5:53	Adam Kie	2	1/7/2002 14:05	Windows	P1	fraenkel	normal	2
7639	JDT	UI	akiezun	VERIFIED	FIXED	NPE when	2/13/2002 8:29	Adam Kie	3	1/15/2002 14:44	Windows	P1	lan_Peter	normal	2
7873	JDT	UI	akiezun	VERIFIED	FIXED	Bogus err	3/28/2002 5:22	Adam Kie	5	1/17/2002 17:37	Windows	P1	lynn_kue	normal	2
8023	JDT	UI	akiezun	VERIFIED	FIXED	coreexcep	2/13/2002 6:29	Adam Kie	6	1/22/2002 12:32	Windows	P1	aeschli	major	2
8129	JDT	UI	akiezun	VERIFIED	FIXED	NPE in Ext	3/28/2002 6:49	Adam Kie	3	1/23/2002 10:12	Windows	P1	aeschli	normal	2
16709	JDT	UI	andre_we	VERIFIED	FIXED	Backgrour	6/4/2002 6:24	Andre We	7	5/21/2002 20:04	Windows	P1	richkulp	blocker	2
16733	JDT	UI	andre_we	VERIFIED	FIXED	Builder pr	6/4/2002 8:23	Andre We	6	5/22/2002 6:17	Windows	P1	philippe_	normal	2
16781	JDT	UI	andre_we	VERIFIED	FIXED	Missing W	6/3/2002 8:11	Andre We	11	5/22/2002 9:44	Windows	P1	daniel_m	trivial	2
16811	JDT	UI	andre_we	VERIFIED	FIXED	J Working	6/3/2002 9:15	Andre We	7	5/22/2002 10:28	Windows	P1	daniel_m	major	2
18807	Platform	Compare	andre_we	VERIFIED	FIXED	Compare v	6/13/2002 5:00	Andre We	12	6/3/2002 13:01	Windows	P1	daniel_m	blocker	2
136087	Platform	Compare	andre_we	VERIFIED	FIXED	Cannot re	4/26/2006 10:20	Andre We	7	4/11/2006 7:04	Windows	P1	jerome_la	critical	3.1
141082	Platform	Compare	andre_we	VERIFIED	FIXED	CompareE	5/12/2006 11:06	Andre We	12	5/10/2006 12:44	All	P1	bokowski	critical	3.2
429301	Platform	SWT	arunkuma	VERIFIED	FIXED	Workspac	3/3/2014 5:14	Arun Thor	10	2/28/2014 5:22	Linux	P1	Lars.Voge	blocker	4.4
151683	JDT	UI	benno.bai	VERIFIED	FIXED	[extract st	5/15/2008 11:02	Benno Bai	11	7/25/2006 8:28	Windows	P1	benno.bai	major	3.2

Figure 5.3 : Sample Dataset

INTEGRATED DEVELOPER ENVIROMENT (IDE):

NETBEANS

NetBeans is a coordinated advancement condition (IDE) for Java. NetBeans enables applications to be created from a lot of measured programming parts called modules. It runs on Windows, macOS, Linux and Solaris. Not with standing Java improvement, it has expansions for different dialects like PHP, C, C++, HTML5, and it's a javascript application dependent on Netbeans, including the Netbeans ide, can be stretched out by outsider designers.

5.4 IMPLEMENTATION:

5.4.1 THE ADMIN MODULE

ADMIN LOGIN:

The developed tool allows admin to enter credentials as shown in Figure 5.4 and upload the dataset as shown in Figure 5.5 to train the program about various bugs with their severity.



Figure 5.4 : Admin Login



Figure 5.5 : Upload Dataset

VIEW DATASET:

Once the admin is logged in and the dataset is uploaded, the same dataset can be viewed by admin as shown in Figure 5.6. Admin can check whether the same dataset has been uploaded correctly or not.

Proposed New Hybrid Solution for Error Severity Improvements

View All Datasets

Total Data's Count : 499

[Click here Preprocessing](#)

Bug_Id	Product	Bugs	Assigned Date	Fixed Date	Assignee	Reporter
140299	JDT	failing Serial Version Clean Up test	5/5/2006 3:15	5/11/2006 7:07	Benno Baumgartner	eclipse
140312	JDT	Occurrences in File quick menu (Ctrl+Shift+U) not available in binary class	5/5/2006 4:23	5/8/2006 11:00	Benno Baumgartner	markus_keller
217790	JDT	[breadcrumb] Remove left/right arrows	2/5/2008 2:50	2/7/2008 4:00	Benno Baumgartner	daniel_megert
217800	JDT	[breadcrumb] Exception when selecting entry from dropdown	2/5/2008 5:02	2/7/2008 5:18	Benno Baumgartner	aeschli
222549	Platform	[navigation] Multi-hyperlink shell at wrong location	3/13/2008 7:19	4/28/2008 4:09	Benno Baumgartner	daniel_megert
223558	JDT	[breadcrumb] Blocked in endless display loop while opening dropdown	3/22/2008 11:06	3/27/2008 12:52	Benno Baumgartner	markus_keller
223586	JDT	[misc] NPE on enumerations in javadoc view	3/23/2008 15:02	3/27/2008 11:37	Benno Baumgartner	benjamin.muskall
33897	Platform	[Import/Export] Export to .tar.gz or .tar.bz2	3/5/2003 14:47	3/1/2005 2:18	Billy Biggs	johan.walles
68033	Platform	oopuu menu has no size in ctrl+e drop down editor list	6/21/2004	6/23/2004	Billy Biggs	michaelvanmeekere

Figure 5.6 : View Dataset

PRE_PROCESSED DATA:

The next step in bug prediction is pre-processing the data using text mining technique which is shown in Figure 5.7.

Proposed New Hybrid Solution for Error Severity Improvements

View Preprocessed Data

Total Data's Count : 499

[Click here Feature Extraction](#)

Bug_Id	Product	Bugs	Assigned Date	Fixed Date	Assignee	Reporter
140299	JDT	fail Serial Version Clean test	5/5/2006 3:15	5/11/2006 7:07	Benno Baumgartner	eclipse
140312	JDT	Occurrences File quick menu Ctrl+Shift+U available binary class	5/5/2006 4:23	5/8/2006 11:00	Benno Baumgartner	markus_keller
217790	JDT	breadcrumb Remove left/right arrows	2/5/2008 2:50	2/7/2008 4:00	Benno Baumgartner	daniel_megert
217800	JDT	breadcrumb Exception select entry dropdown	2/5/2008 5:02	2/7/2008 5:18	Benno Baumgartner	aeschli
222549	Platform	navigate Multi-hyperlink shell wrong locate	3/13/2008 7:19	4/28/2008 4:09	Benno Baumgartner	daniel_megert
223558	JDT	breadcrumb Block endless display loop opening dropdown	3/22/2008 11:06	3/27/2008 12:52	Benno Baumgartner	markus_keller
223586	JDT	misc NPE enumerations javadoc view	3/23/2008 15:02	3/27/2008 11:37	Benno Baumgartner	benjamin.muskall
33897	Platform	Import/Export Export targz tarbz2	3/5/2003 14:47	3/1/2005 2:18	Billy Biggs	johan.walles

Figure 5.7 : Pre-Processing Data

FEATURE EXTRACTION:

Once the pre-processed data is collected, now the important features or words are extracted which is shown in Figure 5.8. This step is known as feature extraction which helps in reducing the overall word count.

Proposed New Hybrid Solution for Error Severity Improvements

Feature Extraction

Total Data's Count : 499 [Click here to TF/IDF Score](#)

Bug_Id	Product	Bugs	Assigned Date	Fixed Date	Assignee	Reporter
140299	JDT	fail Serial Version Clean test	5/5/2006 3:15	5/11/2006 7:07	Benno Baumgartner	eclipse
140312	JDT	Occurrences File quick menu Ctrl+Shift+U available binary class	5/5/2006 4:23	5/8/2006 11:00	Benno Baumgartner	markus_keller
217790	JDT	breadcrumb Remove left/right arrows	2/5/2008 2:50	2/7/2008 4:00	Benno Baumgartner	daniel_megert
217800	JDT	breadcrumb Exception select entry dropdown	2/5/2008 5:02	2/7/2008 5:18	Benno Baumgartner	aeschli
222549	Platform	navigate Multi-hyperlink shell wrong locate	3/13/2008 7:19	4/28/2008 4:09	Benno Baumgartner	daniel_megert
223558	JDT	breadcrumb Block endless display loop opening dropdown	3/22/2008 11:06	3/27/2008 12:52	Benno Baumgartner	markus_keller
223586	JDT	misc NPE enumerations javadoc view	3/23/2008 15:02	3/27/2008 11:37	Benno Baumgartner	benjamin.muskalla
33897	Platform	Import/Export Export targz tarbz2	3/5/2003 14:47	3/1/2005 2:18	Billy Biggs	johan.walles
68033	Platform	popup menu size ctrl+e drop editor list	6/21/2004 12:17	6/23/2004 13:45	Billy Biggs	michaelvanmeekere

Figure 5.8 : Feature Extraction

Calculate TF-IDF Score:

Now the TF and IDF score is calculated based on creation of a term document matrix. TF and IDF values are shown in Figure 5.9. The TF and IDF weight characterized for a word is frequently used in data recovery and content mining.

Proposed New Hybrid Solution for Error Severity Improvements

Overall Rank Score

Total Data's Count : 499

[Click here to Predict Bug Severity](#)

Bug_Id	Product	Bugs	Assigned Date	Fixed Date	TF Value	IDF Value	TF/IDF Score
140299	JDT	failing Serial Version Clean Up test	5/5/2006 3:15	5/11/2006 7:07	1.0	1.6094379124341003	1.609437912434100
140312	JDT	Occurrences in File quick menu (Ctrl+Shift+U) not available in binary class	5/5/2006 4:23	5/8/2006 11:00	1.0	1.791759469228055	1.79175946922805
217800	JDT	[breadcrumb] Exception when selecting entry from dropdown	2/5/2008 5:02	2/7/2008 5:18	1.0	2.302585092994046	2.30258509299404
222549	Platform	[navigation] Multi-hyperlink shell at wrong location	3/13/2008 7:19	4/28/2008 4:09	1.0	1.791759469228055	1.79175946922805
223558	JDT	[breadcrumb] Blocked in endless display loop while opening dropdown	3/22/2008 11:06	3/27/2008 12:52	1.0	2.0794415416798357	2.07944154167983
223586	JDT	[misc] NPE on enumerations in javadoc view	3/23/2008 15:02	3/27/2008 11:37	1.0	1.6094379124341003	1.60943791243410
68033	Platform	popup menu has no size in ctrl+e drop down editor list	6/21/2004 12:17	6/23/2004 13:45	1.0	1.6094379124341003	1.60943791243410
82988	Platform	[Import/Export] Importing Existing Projects should support Archive Files	1/17/2005 11:57	5/10/2005 11:42	1.0	2.0794415416798357	2.07944154167983
90006	Platform	[Progress] CVS repositories view flashes while browsing	4/2/2005	5/10/2005	1.0	1.3863043611108006	1.38630436111080

Figure 5.9 : Calculate TF-IDF Score

View Bugs Severity:

Finally the severity of bug can be viewed as shown in Figure 5.10

Proposed New Hybrid Solution for Error Severity Improvements

Predict Bug Severity Level

Total Data's Count : 395

[Click Here to Performance Analysis](#)

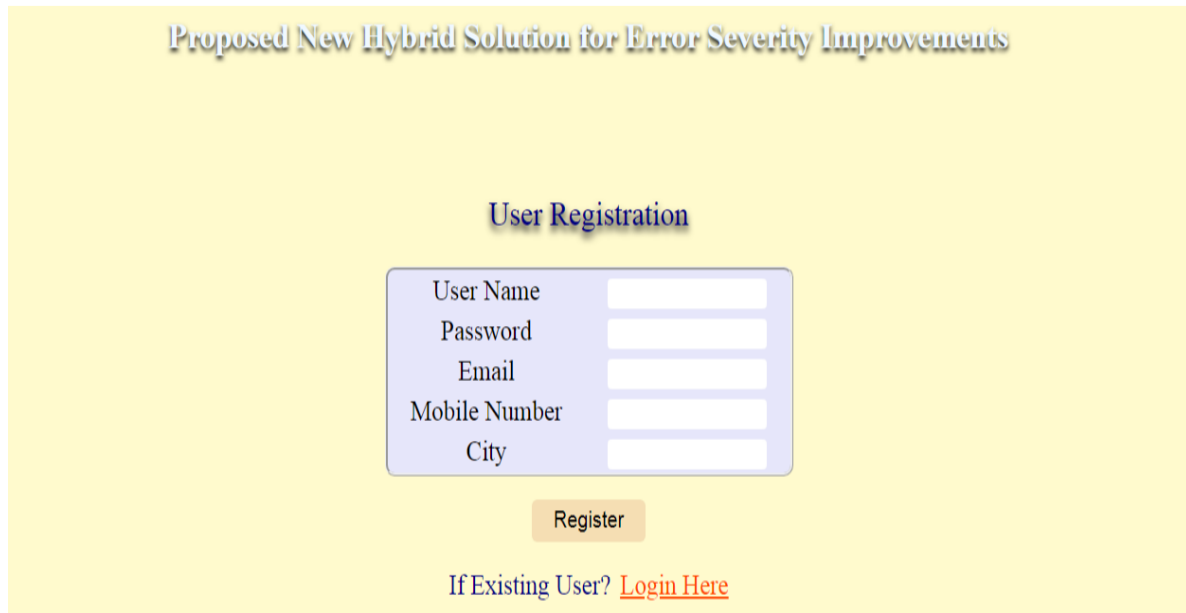
Id	Product	Bugs	Assigned Date	Fixed Date	Assignee	Reporter	Severity Level
99	JDT	failing Serial Version Clean Up test	5/5/2006 3:15	5/11/2006 7:07	Benno Baumgartner	eclipse	major
12	JDT	Occurrences in File quick menu (Ctrl+Shift+U) not available in binary class	5/5/2006 4:23	5/8/2006 11:00	Benno Baumgartner	markus_keller	normal
00	JDT	[breadcrumb] Exception when selecting entry from dropdown	2/5/2008 5:02	2/7/2008 5:18	Benno Baumgartner	aeschli	major
49	Platform	[navigation] Multi-hyperlink shell at wrong location	3/13/2008 7:19	4/28/2008 4:09	Benno Baumgartner	daniel_megert	normal
58	JDT	[breadcrumb] Blocked in endless display loop while opening dropdown	3/22/2008 11:06	3/27/2008 12:52	Benno Baumgartner	markus_keller	critical
86	JDT	[misc] NPE on enumerations in javadoc view	3/23/2008 15:02	3/27/2008 11:37	Benno Baumgartner	benjamin.muskalla	normal
33	Platform	popup menu has no size in ctrl+e drop down editor list	6/21/2004 12:17	6/23/2004 13:45	Billy Biggs	michaelvanmeekeren	normal
88	Platform	[Import/Export] Importing Existing Projects should support Archive Files	1/17/2005 11:57	5/10/2005 11:42	Billy Biggs	aaron	enhancement

Figure 5.10 : View Bug Severity

5.4.2 THE USER MODULE:

The User Registration:

Along with admin login, any user can create credentials as shown in Figure 5.11 and get logged in which is shown in Figure 5.12.



Proposed New Hybrid Solution for Error Severity Improvements

User Registration

User Name	<input type="text"/>
Password	<input type="password"/>
Email	<input type="text"/>
Mobile Number	<input type="text"/>
City	<input type="text"/>

Register

If Existing User? [Login Here](#)

Figure 5.11 : User Registration



Proposed New Hybrid Solution for Error Severity Improvements

User Login

User Name	<input type="text"/>
Password	<input type="password"/>

Login

If New User? [Click Here](#)

Figure 5.12 : User Login

VIEW PROFILE:

User created profile can be viewed by the user as shown in Figure 5.13 which shows all the details entered to be viewed by user.

The screenshot shows the 'View My Profile' page. At the top left is the application logo 'Bug Severity Prediction' in pink. At the top right are navigation links: 'HOME', 'ADD BUGS', 'VIEW SEVERITY LEVEL', and 'LOGOUT'. The main content area has a light orange background with the title 'View My Profile' in orange. Below the title is a table with the following data:

User ID	1
User Name	padmasri
Email	icoreprojects.javateam@gmail.com
Mobile	9988776655
City	chennai

Figure 5.13 : View Profile

ADD BUGS:

Users can add new detected bugs to the database as shown in Figure 5.14 with its severity. This will allow the dataset to increase more in size with different keywords.

The screenshot shows the 'Add New Bugs' form. At the top left is the application logo 'Bug Severity Prediction' in pink. At the top right are navigation links: 'HOME', 'ADD BUGS', 'VIEW SEVERITY LEVEL', and 'LOGOUT'. The main content area has a light orange background with the title 'Add New Bugs' in orange. Below the title is a form with the following fields:

- Bug Name:
- Date:
- Bugs Keyword:
- Bug Description:

At the bottom of the form is an 'ADD' button.

Figure 5.14 : Add Bugs

View Severity Level:

Finally, the severity of the bug can be detected as soon as any bug is entered in the tool. This is shown in Figure 5.15.

Bug Severity Prediction

HOME ADD BUGS VIEW SEVERITY LEVEL LOGOUT

View severity Level of Bugs

Bug_Id	Bugs	Date	Description	Status
1	Package install	19-11-2019	Try to solve as soon as possible	Normal

Figure 5.15 : View Severity Level

CHAPTER 6

RESULTS

After training numerous models we were able to achieve an overwhelming best validation accuracy of 80 % with the proposed solution. As a result this classification allows us to predict the severity of the bug reported. The bugs that are reported after software deployment affect the quality and reliability of the software. After training the model we were able to predict the severity and classify the reported bug in particular category. Figure 6.1 shows the precision and accuracy of the tool developed.

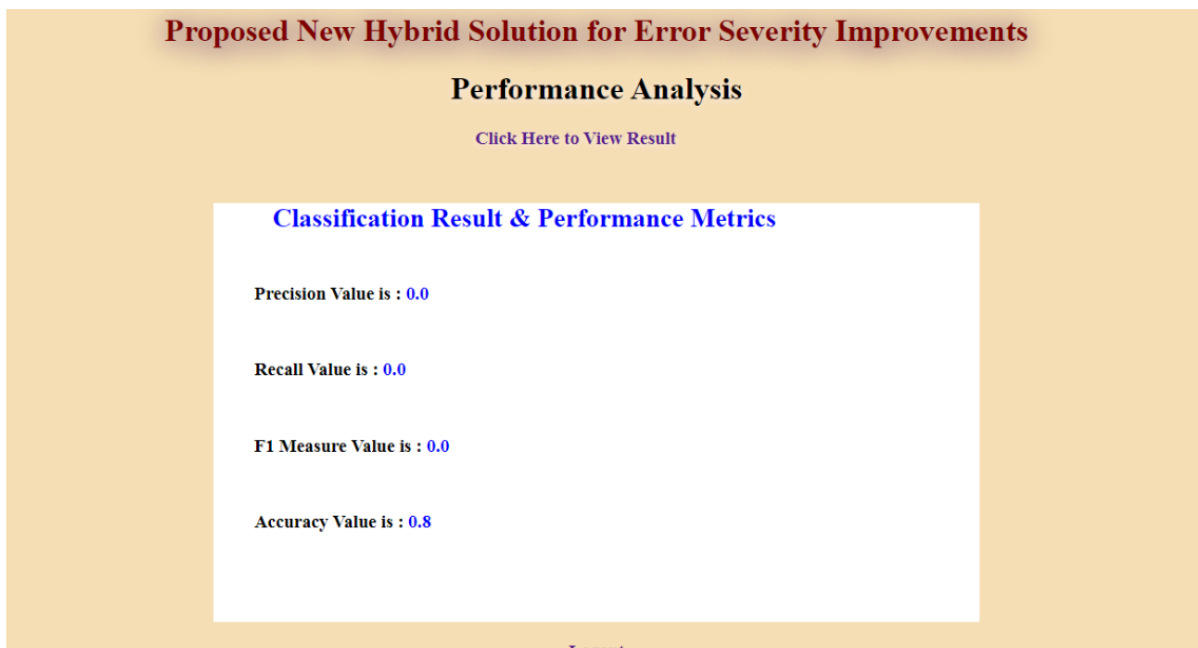


Figure 6.1 : Accuracy of trained model

The output of our tool shows the severity level of collected bugset. Prediction results have been divided in various categories like critical, enhancement, trivial, major, minor, normal as shown in Figure 6.2. Our proposed work depicted the approach to classify and select feature for categorizing the bug into different levels.

Prediction Results

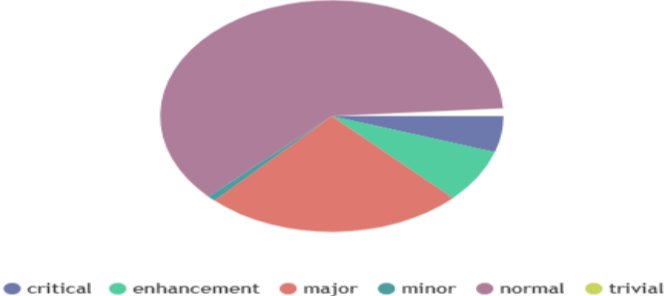


Figure 6.2 : Prediction Results

CHAPTER 7

CONCLUSION AND FUTURE WORK

These product bugs are distinguished after sending programming influence the dependability and nature of programming. Bug following frameworks enable clients to report these bugs of numerous open source programming. Anyway, anticipating the seriousness level of these bug report is rising issue. Various undertakings have been made to address the issue of reality figure, yet no undertaking was made for making word reference of fundamental terms of earnestness marker. The work displayed in this paper proposed a component decision and request approach for arranging the bug reports into outrageous and non-genuine class. Feature assurance systems filter through commonly instructive terms from datasets resulting to preprocessing steps.

FUTURE WORK:

In this exploration, content arrangement for evaluating the seriousness levels of the imperfection reports and foreseeing the seriousness levels of inconspicuous deformity reports continuously has been utilized. In future this examination is expected to utilize this instrument for non-utilitarian prerequisites report and support demand report.

REFERENCES:

- [1] D. Carbamic and G. C. Murphy, "Automatic bug triage using text categorization," in *Proc Sixteenth International Conference on Software Engineering*.
- [2] L. Yu, C. Kong, L. Xu, J. Zhao and H. Zhang, "Mining Bug Classifier and Debug Strategy Association Rules for Web-Based Applications," in *08 Proceedings of the 4th international conference on Advanced Data Mining and Applications*, 2008.
- [3] S. Ahsan , J. Ferzund and F. Wotawa, "Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine," in *Proceedings of the 2009 Fourth International Conference on Software Engineering Advances*, p.216-221, September 20-25, 2009.
- [4] N.Suguna and Dr. K. T. di, "An Improved k-Nearest Neighbor Classification Using Genetic Algorithm," in *IJCSI International Journal of Computer Science Issues*, vol. 7, Issue 4, No 2, July 2010.
- [5] G. Canfora and L. Cerulo, "Impact Analysis by Mining Software and Change Request Repositories," in *Proceedings 11th IEEE International Symposium on Software Metrics (METRICS'05)*, September 19-22 2005, pp. 20-29.
- [6] D. Cubranic and G. C. Murphy, "Automatic Bug Triage Using Text Categorization," in *Proceedings 6th International Conference on Software Engineering & Knowledge Engineering (SEKE'04)*, 2004, pp. 92-97.
- [7] M. P. ILIEV, "A method for automated prediction of defect severity using ontologies," in Master's thesis, LIACS, Leiden University, Logica Netherlands, 2012
- [8] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *IEEE International Conference on Software Maintenance*, 28 2008-Oct. 4 2008, pp. 346-355.
- [9] I. Herraiz, D. German, J. Gonzalez-Barahona, and G. Robles, "Towards a Simplification of the Bug Report Form in Eclipse," in *5th International Working Conference on Mining Software Repositories*, May 2008.

- [10] G. Jeong, S. Kim, and T. Zimmermann, “Improving Bug Triage with Tossing Graphs,” in *Proc. 17th ACM SIGSOFT Symp. Foundations of Software Engineering (FSE '09)*, Aug. 2009, pp. 111-120.
- [11] G. Salton, “Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer,” Addison-Wesley.
- [12] T. Menzies, J. Greenwald and A. Frank, “Data Mining Static Code Attributes to Learn Defect Predictors,” in *IEEE Transactions on Software Engineering*.
- [13] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello, “Extracting Places from Traces of Locations,” in *Proceedings of the 2Nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, ser. WMASH 04*. Philadelphia, PA, USA: ACM, 2004, pp. 110–118.
- [14] J. Reades, F. Calabrese, A. Sevtsuk, and C. Ratti, “Cellular Census: Explorations in Urban Data Collection,” *IEEE Pervasive Computing*, vol. 6, no. 3, pp. 30–38, Jul. 2007.
- [15] G. Rose, “Mobile Phones as Traffic Probes: Practices, Prospects and Issues,” *Transport Reviews*, vol. 26, no. 3, May 2006.