

A Major Project Report On

ASSESSING SEVERITY OF SOFTWARE DEFECT REPORTS USING MACHINE LEARNING TECHNIQUES

Submitted in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY IN SOFTWARE ENGINEERING

By

Aditya Hriday Jalan

(Roll No. 2K12/SWE/01)

Under the guidance of

Dr. Ruchika Malhotra

Department of Software Engineering

Delhi Technological University, Delhi



Department of Computer Engineering

Delhi Technological University, Delhi

2012-2014



DELHI TECHNOLOGICAL UNIVERSITY
CERTIFICATE

This is to certify that the project report entitled **ASSESSING SEVERITY OF SOFTWARE DEFECT REPORTS USING MACHINE LEARNING TECHNIQUES** is a bona fide record of work carried out by Aditya Hriday Jalan (2K12/SWE/01) under my guidance and supervision, during the academic session 2012-2014 in partial fulfilment of the requirement for the degree of Master of Technology in Software Engineering from Delhi Technological University, Delhi.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Dr. Ruchika Malhotra

Asst. Professor

Department of Software Engineering

Delhi Technological University

Delhi



DELHI TECHNOLOGICAL UNIVERSITY

ACKNOWLEDGEMENT

With due regards, I hereby take this opportunity to acknowledge a lot of people who have supported me with their words and deeds in completion of my research work as part of this course of Master of Technology in Software Engineering.

To start with I would like to thank the almighty for being with me in each and every step of my life. Next, I thank my parents and family for their encouragement and persistent support.

I would like to express my deepest sense of gratitude and indebtedness to my guide and motivator, **Dr. Ruchika Malhotra**, Assistant Professor, Department of Software Engineering, Delhi Technological University for her valuable guidance and support in all the phases from conceptualization to final completion of the project.

I wish to convey my sincere gratitude to all the faculties and PhD. Scholars of Computer Engineering Department, Delhi Technological University who have enlightened me during my project.

I humbly extend my grateful appreciation to my friend Anushree Agrawal whose moral support made this project possible.

Last but not the least, I would like to thank all the people directly and indirectly involved in successfully completion of this project.

Aditya Hriday Jalan

Roll No. 2K12/SWE/01

TABLE OF CONTENTS

Certificate	i
Acknowledgement	ii
Table of contents	iii-iv
List of Figures	v-vi
List of Tables	vii
Abstract	viii
Chapter 1: Introduction	1-4
1.1 Introduction	1
1.2 Motivation of Work	1
1.3 Aim of Work	2
1.4 Organization of Thesis	4
Chapter 2: Related Work	5-6
Chapter 3: Research Methodology	7-15
3.1 Flowchart of the methodology	7
3.2 Tokenization	9
3.3 Stop Word Removal	9
3.4 Stemming	10
3.5 Tf-Idf	10
3.6 Info Gain	11
3.7 Multilayer Perceptron	12
3.8 Multi nominal Logistic Regression	13
3.9 k-fold cross Validation	14
Chapter 4: Result Analysis	15-44

4.1 Data Source	15
4.2 PITS C Result.....	17
4.2.1 Data Pre-processing	17
4.2.2 Dimensionality Reduction	18
4.2.3 Generating Tf-Idf Matrix.....	19
4.2.4 k-fold cross Validation	20
4.2.5 Multilayer Perceptron	20
4.2.6 Multi nominal Logistic Regression	23
4.2.7 Comparison of Techniques	26
4.3 TOMCAT Results	27
4.3.1 Data Pre-processing	28
4.3.2 Dimensionality Reduction	28
4.3.3 Generating Tf-Idf Matrix.....	29
4.3.4 k-fold cross Validation	30
4.3.5 Multilayer Perceptron	32
4.3.6 Multi nominal Logistic Regression	35
4.3.7 Comparison of Techniques	38
Chapter 6:- Conclusion and Future Work	39-40
References	41-42

LIST OF FIGURES

3.1 Flowchart describing general methodology	7
3.2 MLP Network with One Hidden Layer.....	13
4.2.1 Initial defect report of PITS C	17
4.2.2 Pre-processed PITS C dataset	18
4.2.3 Tf-Idf matrix for PITS C dataset	19
4.2.4 Tf-idf matrix for PITS C data set as imported in SPSS software	21
4.2.5 ROC curve for Sev. 3 corresponding to ‘val 1’ partition.variable in PITS C dataset.....	22
4.2.6 ROC curve for Sev. 4 corresponding to ‘val 1’ partition variable in PITS C dataset.....	22
4.2.7 ROC curve for Sev. 5 corresponding to ‘val 1’ partition variable in PITS C dataset.....	23
4.2.8 ROC curve for Sev. 3 corresponding to ‘val 7’ partition.variable in PITS C dataset.....	25
4.2.9 ROC curve for Sev. 4 corresponding to ‘val7’ partition variable in PITS C dataset.....	25
4.2.10 ROC curve for Sev. 5 corresponding to ‘val7’ partition variable in PITS C dataset.....	26
4.3.1 Initial defect report of TOMCAT	27
4.3.2 Pre-processed TOMCAT dataset	28
4.3.3 Tf-Idf matrix for TOMCAT dataset	30
4.3.4 Tf-idf matrix for TOMCAT data set as imported in SPSS software	31
4.3.5 Tf-idf matrix for TOMCAT data set with 10 validation parameters	32
4.3.6 ROC curve for Sev.1 corresponding to ‘val 4’ partition.variable in TOMCAT dataset ...	33
4.3.7 ROC curve for Sev. 2 corresponding to ‘val4’ partition variable in TOMCAT dataset ...	34
4.3.8 ROC curve for Sev. 3 corresponding to ‘val4’ partition variable in TOMCAT dataset	34
4.3.9 ROC curve for Sev. 1 corresponding to ‘val 2’ partition.variable in TOMCAT dataset	36

4.3.10 ROC curve for Sev.2 corresponding to 'val2' partition variable in TOMCATdataset ..37

4.3.11 ROC curve for Sev. 3 corresponding to 'val2' partition variable in TOMCATdataset..37

LIST OF TABLES

4.1 Number of documents corresponding to the severity in PITS C dataset	16
4.2 Number of documents corresponding to the severity in PITS C dataset	16
4.2.1 Top 100 words of PITS C dataset based on the frequency.....	19
4.2.2 AUC obtained by ROC after applying 10 rounds of MLP on PITS C dataset	21
4.2.3 AUC obtained by ROC after applying 10 rounds of MLR on PITS C dataset	24
4.2.4 AUC obtained by ROC after applying 10 rounds of MLR and MLP on PITS C data	26
4.3.1 Top 100 words of PITS C dataset based on the frequency.....	29
4.3.2 AUC obtained by ROC after applying 10 rounds of MLP on TOMCAT dataset ...	33
4.3.3 AUC obtained by ROC after applying 10 rounds of MLR on TOMCAT dataset ...	35
4.3.8 AUC obtained by ROC after applying 10 rounds of MLR and MLP on PITS C data	38

ABSTRACT

As per the software development, software testing is one of the most important phases of software life cycle. And similarly, a defect report is a key document which is required for software testing. We need to maintain testing reports and defect reports to keep track of the behaviour of software, whether it is going on as desired or we need to make changes in the undergoing software development. But as the software complexity increases, the number of defects also increases. Our prime focus then relies on looking for the defects and classifying them on the basis of severity.

Severity assessment is of prime focus for test engineers. Actually, most of the defect reports generated by almost any kind of software tool generate a log report. Such log reports contain description of the defects encountered. It is difficult to scan each and every line and find out the severity of the defects. So, there is a need for a system that scans various log reports and classifies it in various categories as low, medium, high on the basis of keywords encountered in the defect report.

The main idea behind this paper can be broadly classified in two heads, text classification and machine learning techniques. As a subject, we have chosen the NASA's Project and Issue Tracking System (PITS) dataset and TOMCAT dataset. Various text classification techniques have been applied to extract raw data from the log report. Then, we have applied machine learning techniques over it to get the severity report. To validate the result, k-fold cross validation method is applied over data in different machine learning techniques. The machine learning technique used here is Multilayer Perceptron and statistical method used is Multinomial Logistic Regression. It has been observed that MLP method has given better results in all of the cases as compared to MLR method.

INTRODUCTION

1.1 Introduction

Over the years, a large number of projects have been done in the field of software engineering. These projects take aid from various concepts underlying in the nature and various subjects relevant in the world. Most of them include the defect reports. We need to work on these defect reports and then analyse the severity of these defect reports on the basis of key words found in them. If we look on to the broader picture, we'll realize that most of the defects and related key words are similar in nature. There is quite a similar or common set of attributes found in the datasets and then related defect reports generated thereafter. There are chances that a type of defect occurring in one type of projects impact the other projects quite high or does not even impact some other defects at all. But each of them needs to be analysed so as to check their impact on the overall system.

Our idea is to analyse each of the defect reports. We study different defect reports, extract the key words (those words which are relevant to the defects and occur most frequently in the report), and then apply machine learning techniques over them to study the impact that these words have on the overall report. Further, we apply validation technique over these words to be sure that these words are actually relevant and we can consider their importance in other projects as well.

1.2 Motivation of the work

There have been many research works going in the field of defect analysis or bug report analysis. The most major problem being is to start. Actually the defect entries do not follow a proper trend in defect report. Most of the defect reports are quite clumsy. They are meant to be in human readable form, rather machine readable form. That's why there is this problem. Many researchers have used machine learning techniques and statistical methods that describe the textual similarities between bugs. The use of techniques like Multinomial regression called for the use of further

different machine learning techniques that could eventually improve the result and be proved to be more effective. The extraction of important words for this job again seemed to be a big task for which data pre-processing technique is used. Stemming proved to be a necessary aid in this step. Finally the result found out to be quite useful, and it even calls for further research works in this area.

1.3 Aim of the work

The basic concept involving this text classification can be broadly classified into different section.

1. Data Pre-processing
2. Dimensionality reduction using InfoGain measure
3. Generating Tf-idf matrix
4. Generating data set for k-fold cross validation using IBM SPSS tool
5. Application of statistical and machine learning techniques

Let's describe the above concepts.

Data Pre-processing:- This method deals with dimensionality reduction or reducing the number of attributes present in the documents. The prime focus of data pre-processing is to remove irrelevant data from the complete defect report. These data include date on which defect was logged, title of defect, nature of defect etc. Having left with only the defect description, we remove the irrelevant words from the report. These words comprise of grammatical word forms as pronouns, adverbs, adjectives, prepositions, conjunctions etc. A lot of punctuation sign are also incorporated to make the defect report meaningful. All such special symbols and characters are removed. Various abbreviations, numbers etc. are also removed from the dataset. Lastly, we perform the stemming process on this data.

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form – generally a written word form. For example, “run”, “runs”, “ran”, and “running”, are all forms of the same root, conventionally written as “run” and the role of a stemmer is to attribute all the derived forms to the root of the

lexeme. So, this technique deals with removal of all such words and just generating root word of the significant words.

Dimensionality reduction using info gain measure:- After pre-processing, the report contains only the relevant words. However, these words are more than thousands in number. We don't want all these words to be worked on at priority. We remove all the irrelevant words and we are left with a set of relevant attributes from all the defect reports under consideration. These attributes are the root words that describe the nature of the defects. Info gain is a statistical tool that describes the importance of each word in the document. It takes into account the frequency of each word and its corresponding severity at each occurrence and accordingly computes an equivalent real number which forms the basis of ranking the important words.

Generating a Tf-idf matrix:- After info gain, we get the list of all the top root words which are to be analysed. Tf-idf is another parameter that describes the importance of a root word in each lines, unlike info gain that measures its importance in complete report. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. If there be Words number of document and each word i appears Word[i] number of times inside a set of Documents and Document[i] be the documents containing i , then:

$$\mathbf{Tf*idf = Word[i]/Words*log(Documents/Document[i])}$$

Generating data set for k-fold cross validation using IBM SPSS tool:- In this process, the overall matrix is imported in IBM SPSS tool and we use k-fold cross validation for dividing the dataset into training and testing samples randomly. We have used Bernoulli's Random variable with a probability of 0.7 which randomly divides the data into two parts, 70% being training data and 30% being testing data. We have used k=10 in our project. In this way 10 different dividing parameters have been created so that we could apply machine learning techniques 10 times to validate the authenticity of

the result.

Application of statistical and machine learning techniques:- We have used machine learning method Multilayer Perceptron and statistical method Multinomial Logistic Regression for the analysis. The matrix is passed into the IBM SPSS environment. Both the above methods are applied for 10 times corresponding to each partitioning variable. Further, we have divided the datasets based on their sizes. In other words, using text mining process we have obtained different datasets consisting of top 25 words, top 50 words and top 100 words. Each of the matrix is analysed using above techniques and the Area under the curve (AUC) generated by ROC curve for each technique is considered as the performance measure.

1.4 Organisation of the thesis

This thesis report is divided into different chapters. To start with, the first chapter which is the introductory chapter which defines what exactly the problem is and broadly explains the processes which are used to solve the process. The next chapter following introduction is Related Work. This chapter specifies the works done by various authors in this area with special emphasis on the paper “Automated Severity Assessment of Software Defect Report”[1], by Tim Menzies and Andrian Marcus that forms to be the base paper behind this report.

The next chapter is “Research Background” which explains the underlying concept behind each and every technology that we have used in our project demonstrating the relevance of that particular technology. Following this, in the next chapter we explain the “Research Methodologies” that are used in this project. In this section we’ll be discussing how these techniques have been applied in our project. Next we have the “Result Analysis” chapter that describes the implementation section of the project as well as the results obtained using different techniques.

Last but not the least, we provide the conclusion and the future work related to this project which could be taken as a subject to be worked upon.

RELATED WORK

There have been much works in the field of text classification using machine learning techniques and statistical techniques for defect analysis. We will be discussing some of the works that we overviewed.

In the paper by John Anvik et.al [3], they have followed machine learning techniques in two phases. Firstly, they studied the general behaviour of all the defects and the expertise of developers in different types of defects. Finally, they allocated the defect to the best developer for that case. In this way, their aim relied on defect allocation to developers.

In the next paper by G. Canfora et.al [4], they emphasized on the indexing system of Repositories like Bugzilla and CVS. They used the indexing of the software to allocate the defect to best developer.

Third paper we read is authored by G. Canfora et.al [5]. They have used the information retrieval algorithm for deriving the set of source files that are impacted by a proposed change request in above said software repositories.

In a similar paper the authors have used machine learning techniques to study the nature of defects [6]. Their focus was what should be done, when a defect arrives. They applied Bayesian Learning technique over a large number of open source software projects to predict what type of defect it is and which developer it should be assigned to.

Another paper is focused on routing of a maintenance request or service ticket as soon as it is opened [7]. The authors have used Machine Learning techniques like Decision Tree, Bayesian Networks, k-mean clustering to route the service tickets to the correct servicing teams. The results were encouraging with as good as 84% of defects

correctly classified.

There is yet another paper related to our research work where the authors focus on aiding the triagers to take decision whenever a new defect arises [8]. Actually, in case of a new defect, the defect is compared with the other bugs in the open source software bug reports. If it matches to any existing defect, then the defect is compared to be similar and it is discarded. If it doesn't match with any defect and shows a completely different behaviour, then it is added into the bug repository. The authors used different techniques that helped in improvement of result by detecting higher number of duplicate bugs in the repositories than detected earlier.

The paper by Tim Menzies and Andrian Marcus [1] forms the basis of this report. Up till now, all the reports have their focus on defect classification, i.e what to do when a new defect arrives and how to compare whether this defect is a duplicate of existing defect or not. But in this paper, the authors talk about the severity assessment of defect report. Let's talk about this in details.

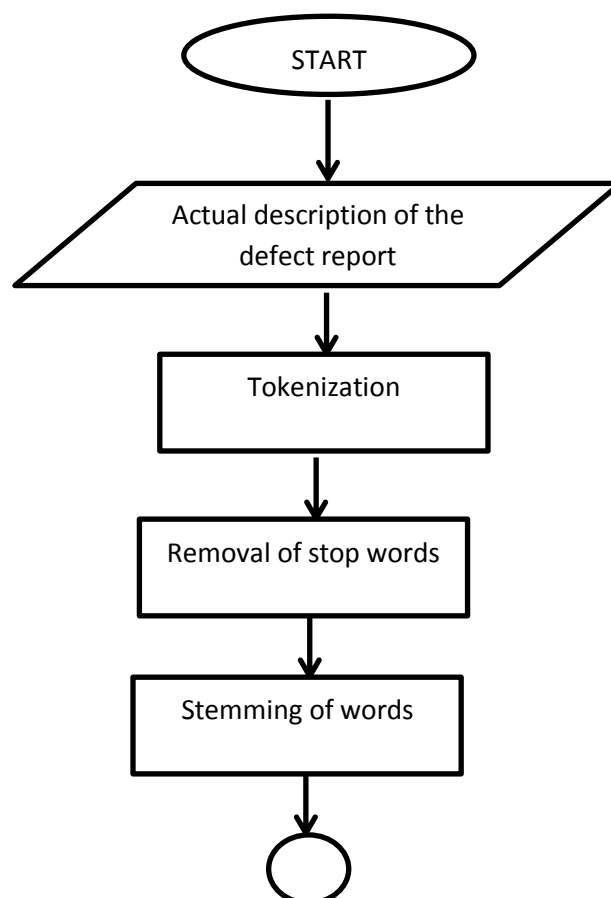
Out of a large number of projects done and available online, we have chosen the defect report of NASA for our subject interest. NASA has developed this system of Project and Issue Tracking System (PITS) wherein they have collected various defects from different projects done over a period of years [2]. Similarly, we have various open source defect reports which are used for various projects. But as of now, we don't have many tools over assessment of these defects found in the reports on the basis of various parameters. We work on this idea proposed by Tim Menzies and Adrian Marcus in their work SEVERIS which is SEVERity ISsue assessment. As per the concept, we will be working in phases for assessing the defect of a defect report. These phases are elaborated in the subsequent chapters.

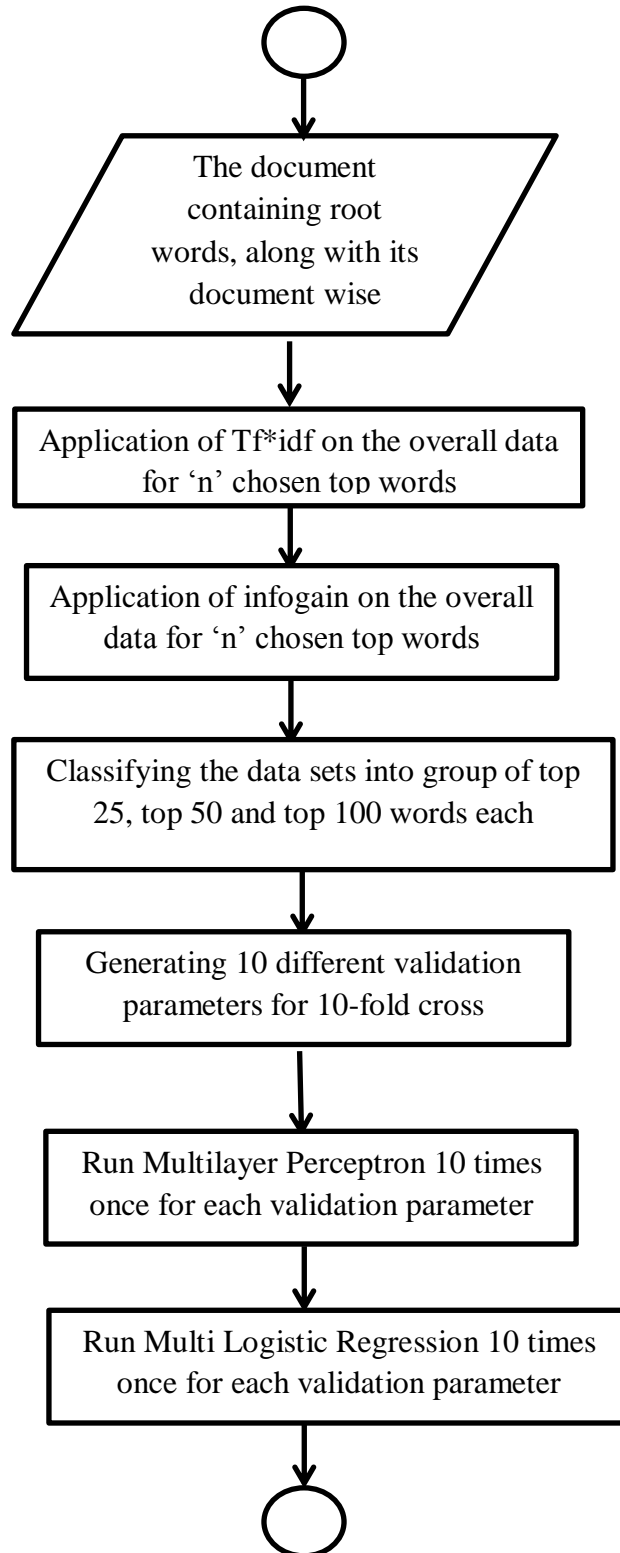
RESEARCH METHODOLOGY

We have come across several machine learning and statistical concepts while working on this project. The text classification concepts were basically used in the pre-processing phase, wherein we extracted useful information from the complete report. The machine learning and statistical models were used in the post processing phase to evaluate the results. In this section, we describe in details about these concepts in the sequence they are used.

Given below, is the flowchart of the overall process and then following it is the complete description of different steps used in the methodology.

3.1 Flowchart of the methodology





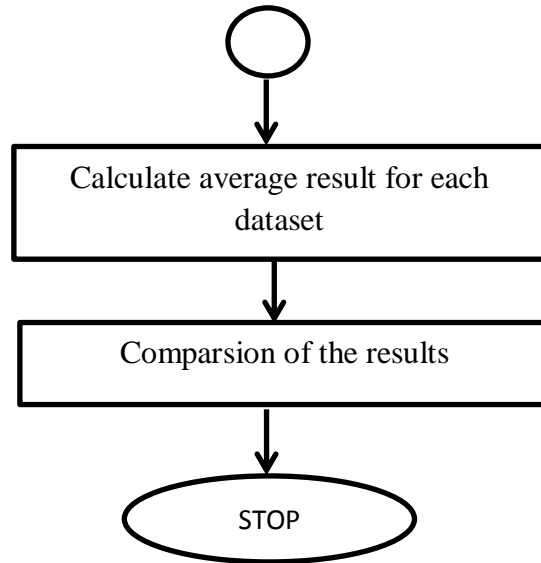


Figure 3.1:- Flowchart describing general methodology

3.2 Tokenization

Tokenization deals with removing of all those extra characters which aids us in sentence formation. For example there are all sorts of punctuation marks, as comma, full stop, brackets, apostrophes, hyphen, dash, quotes, exclamation mark, question mark etc. Further there may be many mathematical operators as addition, multiplication, subtraction symbols, slash, tilde, carat etc. Again, there may be other special symbols as dollars, hash etc. This is done by the tokenization process. Many a times there may be different ASCII symbols present in the document which are not defined in any language. Such symbols are also removed in the process of tokenization. In all we need to get rid of all such special symbols so that our report is left with only English words with no symbols.

3.3 Stop word removal

These are the words which are required to frame sentences. They are not the actual words, but they assist other words for semantics. Few examples of such words are noun, pronoun, verbs, adjectives, adverbs, conjunction, preposition, articles, modals, counting words etc. Along with these words there are special symbols as punctuation marks, bullet symbols, special characters etc. these all need to be removed as these are not the key words. They just aid in sentence formation for the defect report.

There are many online libraries where we find list of such words. For our reference, we have chosen the stop words from <http://norm.al/2009/04/14/list-of-english-stop-words/>.

3.4 Stemming

Stemming of words is the process of extracting the root word from the words. Actually, whenever we make sentences we transform the words to make it more meaningful. This transformation includes applying prefix or suffix to the words. Apart from this the root words are modified to different words as per the tense or the five different forms of a verb. So, our aim relies on transforming these words back to their original root words.

For example, for speak, spoke, spoken, spokes, speaking we will replace with a single word speak. For words like, transferable, applicable, moveable we will remove the suffix able from them. Similarly, there are many words with prefix, 'un', 'in', 'im' etc. These need to be removed. Just like 'able', there is a long list of such words as 'tion', 'te', 'tional' etc. which are removed in the process.

While removing the words, we need to be careful as if remove these words directly, it may lead to removal of many genuine words. So, if we are removing them from the end, we have to check clearly that it doesn't impact the similar letters which are in between the words forming the root word.

3.5 Tf-Idf

Tf-Idf is defined as term frequency times inverse document frequency. This concepts aims to define how much important a particular word is to the entire set of information in the document. In his paper, Juan Ramos[10][17] says, "TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the words appears in."

This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the

number of times a word appears in the document but is offset by the frequency of the word in the corpus. The term frequency in the given document is simply the number of times a given term appears in that document. This count is usually normalized to prevent a bias towards longer documents (which may have a higher term frequency regardless of the actual importance of that term in the document) to give a measure of the importance of the term within a particular document. The inverse document frequency is a measure of the general importance of the term (obtained by dividing the number of all documents by the number of documents containing the term, and then taking the logarithm of that quotient). If there be Words number of document and each word i appears Word[i] number of times inside a set of Documents and if Document[i] be the documents containing i , then:

$$\mathbf{Tf*idf} = \mathbf{Word[i]/Words*log(Documents/Document[i])}$$

The standard way to use this measure is to cull all but the k top $\mathbf{tf*idf}$ ranked stopped, stemmed tokens. The case study presented later in the paper used $k = 100$. The idea is that these are the most important terms for each document, whereas the rest of them can be ignored in the analysis. Mathematically, we can elaborate as:-

$$\mathbf{Tf*idf} = \sum_{i=1}^{\#words} \sum_{j=1}^{\#documents} \# \mathbf{word\ } i \mathbf{ in\ document\ } j * \mathbf{Z\ log_2 Z}$$

$$\text{Where } \mathbf{Z} = \frac{\mathbf{total\ \# documents}}{\# \mathbf{times\ word\ } i \mathbf{ occur\ in\ report}}$$

3.6 Infogain

Info gain is the measure of choosing the word with most severity in the report on the basis of severity assigned to different documents present in the report.

There is an important parameter that is to be found out before info gain. The parameter is entropy. The entropy is defined as a measure of uncertainty in a random variable. It is given as the expected value of all the information given in the message. The general formula that we have used is:-

$$\text{entropy} = - \sum_{i=1}^n \left(\frac{\# \text{ documents with severity } i}{\text{total } \# \text{ documents in report}} \right) \log_2 \left(\frac{\# \text{ documents with severity } i}{\text{total } \# \text{ documents in report}} \right)$$

i.e, it is given by summation of the log (base 2) of the ratio of the number of documents that are assigned to a particular severity and the total number of documents present in the defect report. Once, we are done with the entropy, we calculate the info gain of the system. Info gain is calculated word wise, or it is the info gain for each and every relevant word that comes across the report irrespective of the document in which it lies. Info gain is a parameter that defines the weightage of a word in the report. It defines that which word can best represent the overall meaning of the defect and on the basis of its relevance the words are assigned weights. After info gain is found out, we sort the words on the basis of info gain and choose top words as per the requirement and proceed to find the Tf*idf of such words throughout the report. This Tf*idf forms the basis of machine learning techniques to be applied over data.

$$\text{Infogain} = \text{Entropy} - \left(\sum_{i=1}^{\# \text{ words}} \frac{\# \text{ word } i \text{ in report}}{\# \text{ docs in report}} * \sum_{i=1}^{\# \text{ words}} \sum_{j=1}^{\# \text{ severity}} X \log_2 X \right)$$

3.7 Multi Layer Perceptron

Multilayer Perceptron (MLP) is an example of an artificial neural network. It is used for solving different problems, example pattern recognition, interpolation, etc. It is advancement to the perceptron neural network model. With one or two hidden layers, they can solve almost any problem. They are feed forward neural networks trained with the back propagation algorithm. Error back-propagation learning consists of two passes: a forward pass and a backward pass. In the forward pass, an input is presented to the neural network, and its effect is propagated through the network layer by layer. During the forward pass the weights of the network are all fixed. During the backward pass the weights are all updated and adjusted according to the error computed. An error is composed from the difference between the desired response and the system output. This error information is fed back to the system and adjusts the system parameters in a systematic fashion (the learning rule). The process is repeated until the performance is acceptable [12].

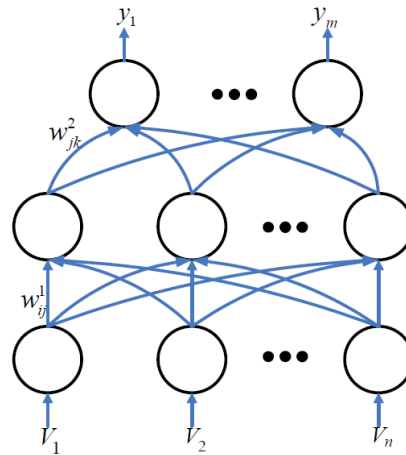


Figure 3.2: MLP Network with One Hidden Layer

3.8 Multi nominal Logistic Regression

Multinomial logistic regression is a statistical classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes. That is, it is a model that is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable [13].

Multinomial Logistic Regression is different from other Logistic Regression as Univariate LR or Multivariate LR. Unlike Univariate LR, there are more than two classes or variables for dependent variables. Hence, it is called multi. One particular feature that discriminates it from Multivariate LR is that in this technique there is an order or sequence of relation between the values of dependent variables. While in later technique, each of the classes are independent to each other having no relation in between.

The analysis breaks the outcome variable down into a series of comparisons between two categories. E.g if we have three outcome categories (A, B and C), then the analysis will consist of two comparisons that we choose:

- Compare everything against the first category (e.g. A vs. B and A vs. C),
- Or the last category (e.g. A vs. C and B vs. C),
- Or a custom category (e.g. B vs. A and B vs. C).

3.9 k-fold cross validation

Validation is the process of checking whether the statistical machine learning techniques that we have applied is actually acceptable or not. The validation process can involve analysing the goodness of fit of the regression, analysing whether the results are random, and checking whether the model's predictive performance deteriorates substantially when applied to data that were not used in model estimation.

We have used cross validation method for validation. This method models validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds. In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data [19]. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data.

We have used Bernoulli's Random function with a probability value of 0.7. We have run this process for 10 times which has created a value of 0 or 1 for each row of dataset in the ratio of 7:3 i.e 70% data is training data and 30% data is testing data. In this way it helps to run the same data set 10 times for each technique but with the difference that the combination of data in training set or testing set is completely random and different from each other.

RESULT ANALYSIS

In this chapter we will be discussing the results obtained in our project. Section 4.1 discusses the different PITS and TOMCAT datasets from where we have chosen data for our case study. In section 4.2, we have chosen PITS C dataset as a case study and in its subsequent sections we have described all the concepts applied on dataset with its results. Similarly, in Section 4.3, we have chosen TOMCAT dataset as a case study and in its subsequent sections we have described all the concepts applied on dataset with its results. Lastly, we have given a comparative analysis of the results obtained by MLP and MLR technique in both the datasets.

4.1 Data Source

There are various sources of defect reports. PITS dataset from NASA (PROMISE repository) is one of the most reputed data source where the scientist have collected data sets from different projects over a span of decades [2]. They have divided the data sets in six different labels from PITS A to PITS F. These data are collected from various scientific and research projects. The data in each of these labels is classified into five severity levels viz. severity 1 to severity 5. These severities are defined as :-

Severity 1:- Very high

Severity 2:- High

Severity 3:- Medium

Severity 4:- Low

Severity 5:- Very Low

We have considered PITS C data for our case study. There are no records corresponding to Severity 1 and Severity 2 in PITS C dataset as can be seen from table 4.1. This is so because the defects falling under the category of level 1 and 2 are of very

high severity levels. Therefore, they are very rare to occur. So, we work upon the data sets for severity 3 to 5. Table 4.1 gives the number of documents corresponding to severity in PITS C data sets.

Project Name	Severity	No. of Documents
PITS C	1	0
	2	0
	3	132
	4	180
	5	7

Table 4.1 Number of documents corresponding to the severity levels in PITS C dataset.

TOMCAT dataset is another dataset that we have taken in account. This dataset also contains similar defect report with defect description, severity and version of the defect. In TOMCAT dataset, the data is broadly classified into the following three severity levels.

Severity 1:- High Impact

Severity 2:- Medium Impact

Severity 3:- Low Impact

There are a total of 178 records in this dataset classified into the three severity levels. Table 4.2 gives the number of documents corresponding to severity in TOMCAT data set.

Project Name	Severity	No. of Documents
TOMCAT	1	73
	2	22
	3	85

Table 4.2 Number of documents corresponding to the severity levels in TOMCAT dataset.

4.2 PITS C Results

Initially, the defect report data is an XL sheet that contains the complete description of defects having different information such as Project_ID, Date, Title, Description, Severity etc. We just choose description out of it, as this column contains the main information about the defect. We apply the data pre-processing technique on this column.

TIM_Id	Subject	Severit	Description	Initiation_t
ProjectC - TIM - 1323	Erroneous switch statement (Build 2, File	3	In Build 2, file CINIT.c, line 119, th	3/13/07
ProjectC - TIM - 1322	Unchecked function return values	3		03-01-2007
ProjectC - TIM - 1321	switch statement has no default (Build 2, F	4	In Build 2, file genport.c, line 257,	03-01-2007
ProjectC - TIM - 1320	Variable not initialized before use (Build 2	3	In Build 2, file mu_init.c, symbol '	1/22/07
ProjectC - TIM - 1319	Boolean within 'if' always evaluates to Tru	3	In Build 2, file schedule.c, lines 70	1/22/07
ProjectC - TIM - 1318	Boolean within 'if' always evaluates to Tru	3	In Build 2, file rd_task.c, the Bool	1/22/07
ProjectC - TIM - 1317	Possible access of out-of-bounds pointer (3	In Build 2, file read_a2d.c, line 19	1/22/07
ProjectC - TIM - 1316	Thermal Testing of Components Not Done	3	In Glory I&T Test Plan document t	1/22/07
ProjectC - TIM - 1315	Reference to Perf. Spec. Rqt 6.4.20 Is Inacc	3	During the review of the ACS Test	1/22/07
ProjectC - TIM - 1314	Inconsistent Roll/Pitch/Yaw Accuracy Valu	3	During an analysis of the ACS Test	1/22/07
ProjectC - TIM - 1313	Wording Differs in Traced Rqt to MU FSW 2	4	MU FSW Section 2.2.6.3 states: "R	1/22/07
ProjectC - TIM - 1312	Wording Differs in Traced Rqt to MU FSW 2	4	MU FSW Section 2.2.6.2 states: "E	1/22/07
ProjectC - TIM - 1311	Wording Differs in Traced Rqt to MU FSW 2	4	MU FSW Section 2.2.6.1 states: "A	1/22/07
ProjectC - TIM - 1310	Wording Differs in Traced Rqt to MU FSW 2	4	MU FSW Section 2.2.6 states: "Th	1/22/07
ProjectC - TIM - 1309	Incomplete Trace for SOC 3.7.1 of TIM Rqts	4	SOC Section 3.7.1 states: "All soft	1/22/07
ProjectC - TIM - 1308	Incomplete Trace for TIM 3.6 of TIM Rqts T	4	TIM Section 3.6 of TIM Rqts Trace	1/22/07
ProjectC - TIM - 1307	Incomplete Trace for MU FSW 1.3.1.4 of TIF	4	MU FSW Section 1.3.1.4 of TIM Rq	1/22/07
ProjectC - TIM - 1306	Incomplete Trace for MU FSW 1.3.1.3 of TIF	4	MU FSW Section 1.3.1.3 of TIM Rq	1/22/07
ProjectC - TIM - 1305	Incomplete Trace for MU FSW 1.3.1.2 of TIF	4	MU FSW Section 1.3.1.2 of TIM Rq	1/22/07
ProjectC - TIM - 1304	Incomplete Trace for MU FSW 1.3.1.1 of TIF	4	MU FSW Section 1.3.1.1 of TIM Rq	1/22/07
ProjectC - TIM - 1303	Incorrect Trace for MU FSW 7.2.2 of TIM Rq	4	MU FSW Section 7.2.2 states: "The	1/22/07
ProjectC - TIM - 1302	Incorrect Trace for MU FSW 7.2 of TIM Rqts	4	MU FSW Section 7.2 states: "After	1/22/07
ProjectC - TIM - 1301	Incorrect Trace for MU FSW 7.2.1 of TIM Rq	4	MU FSW Section 7.2.1 states: "TIN	1/22/07

Figure 4.2.1 Initial defect report of PITS C

4.2.1 Data Pre-processing

Data pre-processing is done using various macros designed in the text pad editor. And then, we achieve a pre-processed data which contains only the root words or the key words that are part of the defect report. After data pre-processing, we get the stemmed text file containing only the key words. This process has to be done with utmost care. Better the stemmed file result, better will be the output after applying machine learning techniques.

```

build file genport switch statement default issu flexelint code analysi tool switch cmdreg case sccl1a comm
build file mu init symbol warm boot signatur initi condit statement issu flexelint code analysi tool exter
build file schedul line ident nest condit evalu issu flexelint code analysi tool command size continu proc
build file task boolean evalu true issu flexelint code analysi tool word return status return return statu
build file read a2d sourc access outofbound pointer condit statement channel larger maximum length a2d loo
glori test plan document thermal test mention section vde propuls integr batteri integr reaction wheel inte
review ac test plan dnprojectcac036 rev5 valu valid test plan test case mention tim1314 pitch roll yaw revi
analysi ac test plan dnprojectcac036 discov inconsist valu rollpitchyaw accuraci point requir document ps s
mu fsw state reject hazard comand enabl trace glori spacecraft tim system icd requir word item list requir
mu fsw state enabl disabl command trace glori spacecraft tim system icd requir word item list requir differ
mu fsw state accept hazard command valid opcod preced enabl command trace glori spacecraft tim system icd r
mu fsw state mu requir enabl hazard command trace glori spacecraft tim system icd requir word item list req
soc state softwar calibr data scienc data product maintain strict configur control trace deriv trace specif
tim tim rqts traceabl matrix document state time sync mode code trace deriv trace specif requir
mu fsw tim rqts traceabl matrix document state powerdown warn messag trace comm icd trace specif requir
mu fsw tim rqts traceabl matrix document state time messag trace comm icd trace specif requir
mu fsw tim rqts traceabl matrix document state ac status messag trace comm icd trace specif requir
mu fsw tim rqts traceabl matrix document state time sync mode code trace comm icd trace specif requir
mu fsw state tim remain safe mode initi spacecraft receipt specif command spacecraft transit mode trace mu
mu fsw state safe mode command spacecraft mu maintain powerdownsaf configur command mode spacecraft trace m
mu fsw state tim oper mode inhibit safe command spacecraft place safe mode trace mu fsw list trace smrd req
mu fsw tim rqts traceabl matrix document state instrument scienc telemetri time stamp error equal second re
mu fsw tim rqts traceabl matrix document state mu ensur timetag synchron sc time trace deriv trace specif r
tps control tim rqts traceabl matrix document state tps support inert target mode joint feedback knowledge
tps control tim rqts traceabl matrix document state tps support solar autotrack mode fss close loop control
mechan state compon mass shall2 measur accuraci lb trace mechan list trace smrd requir exist smrd trace app
mechan state measur tim mass tim launch configurationwith tim panel shall5 perform prior shipment tim syste
mechan state tim mass shall4 base margin mass trace mechan list trace smrd requir exist smrd trace appear i
mechan state mass tim system shall3 includ hardwar reflect mid trace mechan list trace smrd requir exist s
tim state quad radiomet intern baffl block incid light degre half cone angl trace tim list trace deriv smrd
mechan state timglori test milstd461e list tabl trace icd std461e spacecraft tim system icd refer std461c s
mechan state minimum tim glint free tps sun track durat minut trace requir

```

Figure 4.2.2 Pre-Processed PITS C data set

4.2.2 Dimensionality Reduction

After the stemming process, we list the top 25, 50 and 100 words and focus on all such words for the further course of action. A list of top- 100 words sorted on the basis of their frequency corresponding to PITS C dataset is given below. The classification techniques are yet to be applied on these words.

Require	Mu	configur	communic	Sband
State	Data	scienc	detail	Case
Fsw	Cdh	list	launch	Design
Command	System	packet	power	Mention
Specif	Referenc	refer	collect	Asec
Trace	Mode	ap	soh	Cloud
Tim	Artefact	control	card	Maintain
Smrd	Pip	mechan	point	Moc
Perform	Instrument	interfac	rout	Revis
Glory	Verif	adac	compon	Safe
Ground	Initi	channel	realtim	Attitude

Sc	Provid	includ	subsystem	Flight
Document	Traceabl	rate	virtual	Maximum
Spacecraft	Oper	comm	fpga	Degree
lcd	Child	orbit	mission	Electr
Spec	Satellite	safehold	time	Enter
Softwar	Capabl	store	transmitt	Exit
Parent	Rqts	test	ac	Normal
Telemetry	s919er2342	downlink	accuraci	Condit
Matrix	Valid	tps	addit	Error

Table 4.2.1 Top 100 words of PITS C dataset based on the frequency

4.2.3 Generating Tf-Idf matrix

We have used C language to program the above concepts. The file does all the processing and it writes the results in different txt files. The output are two different text files, that contains the infogain values and the Tf*idf values of these words. As per the data, the documents are divided into three severity ratings ranging from 3 to 5, in which the severity rating of 3 means that the defects have a moderate impact and severity rating of 5 means that the defects have a least impact. We have generated a '100 * 323' matrix of data which contains the Tf*idf values of all the above 100 words document wise. The values are then imported to the IBM SPSS tool for further validation using MLP and MLR technique.

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
D1	0	0	0	0	0	0	0	0	0	0	0	0
D2	0	0	0	0	0	0	0	0	0	0	0	0
D3	0	0	0	4	0	0	0	0	0	0	0	0
D4	0	0	0	0	0	0	0	0	0	0	0	0
D5	2	0	0	1	0	0	0	0	0	0	0	0
D6	0	0	0	0	0	0	0	0	0	0	0	0
D7	0	0	0	0	0	0	0	0	0	0	0	0
D8	2	0	0	0	1	0	1	0	1	1	0	0
D9	5	2	0	0	2	0	0	0	3	3	0	0
D10	9	0	0	0	1	0	0	4	1	0	0	0
D11	2	1	1	0	0	2	1	0	0	1	0	0
D12	2	1	1	1	0	2	1	0	0	1	0	0
D13	2	1	1	2	0	2	1	0	0	1	0	0
D14	3	1	1	1	0	2	1	0	0	1	0	0
D15	1	1	0	0	1	2	0	0	0	0	0	0
D16	1	1	0	0	1	2	2	0	0	0	0	0
D17	1	1	1	0	1	2	1	0	0	0	0	0
D18	1	1	1	0	1	2	1	0	0	0	0	0
D19	1	1	1	0	1	2	1	0	0	0	0	0
D20	1	1	1	0	1	2	1	0	0	0	0	0

Figure 4.2.3 Tf-idf matrix for PITS C data set

4.2.4 k-fold cross validation

The Tf-idf matrix is imported to the IBM SPSS tool. We have used Bernoulli's Random function with a probability value of 0.7. We have run this process for 10 times which has created a value of 0 or 1 for each row of dataset in the ratio of 7:3 i.e 70% data is training data and 30% data is testing data. In this way it helps to run the same data set 10 times for each technique but with the difference that the combination of data in training set or testing set is completely random and different from each other.

	Severity	validate1	validate2	validate3	validate4	validate5	validate6	validate7	validate8	validate9	validate10	MLP_Pseudo Probability_1	MLP_Pseudo Probability_2
.00	3	1.00	.00	1.00	1.00	1.00	1.00	1.00	.00	1.00	1.00	.929	.063
.00	3	1.00	.00	1.00	1.00	1.00	.00	1.00	1.00	1.00	1.00	.792	.190
.00	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.930	.062
.00	3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.918	.074
.00	3	1.00	.00	1.00	.00	1.00	1.00	1.00	1.00	.00	1.00	.928	.064
.00	3	1.00	.00	1.00	1.00	1.00	1.00	1.00	1.00	.00	.00	.792	.190
.00	3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.806	.189
.00	3	1.00	1.00	.00	1.00	1.00	1.00	1.00	1.00	1.00	.00	.930	.062
.53	3	1.00	1.00	1.00	1.00	1.00	.00	1.00	1.00	1.00	1.00	.840	.155
.75	3	1.00	1.00	1.00	1.00	1.00	.00	.00	1.00	1.00	1.00	.814	.182
.00	4	1.00	1.00	1.00	1.00	.00	.00	.00	1.00	.00	1.00	.019	.963
.00	4	1.00	1.00	1.00	1.00	.00	1.00	.00	.00	1.00	1.00	.028	.939
.00	4	1.00	1.00	.00	.00	1.00	1.00	.00	.00	1.00	.00	.033	.932
.00	4	1.00	1.00	1.00	1.00	1.00	.00	1.00	1.00	1.00	.00	.031	.930
.00	4	.00	1.00	.00	.00	1.00	1.00	.00	1.00	1.00	1.00	.402	.568
.00	4	.00	.00	1.00	1.00	.00	1.00	1.00	.00	.00	1.00	.091	.846
.00	4	1.00	1.00	.00	1.00	.00	1.00	1.00	1.00	1.00	.00	.049	.889
.00	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.046	.891
.00	4	.00	1.00	.00	1.00	1.00	1.00	.00	1.00	1.00	1.00	.045	.893
.00	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.045	.893
.00	4	1.00	.00	.00	1.00	1.00	1.00	.00	.00	.00	1.00	.014	.973

Figure 4.2.3 Tf-idf matrix for PITS C data set with 10 validation parameters

4.2.5 Multi layer Perceptron

On the above data we have applied multilayer perceptron (MLP) technique 10 times, once for each validation parameter. Initially, we select the data by setting the validate parameter value = 1. After the training data is selected we apply the MLP. Here the top 100 words act as the covariates, the severity acts as the dependent variable and

the validate parameter acts as the partitioning variable. Lastly, we save the predicted value or category for each variable that is used to analyse the result through ROC curve. Table 4.2.2 displays the results obtained after each round of validation for the above mentioned severities.

Runs	Severity 3	Severity 4	Severity 5
Validate 1	0.980	0.982	0.838
Validate 2	0.991	0.996	0.945
Validate 3	0.989	0.993	0.902
Validate 4	0.988	0.986	0.822
Validate 5	0.989	0.992	0.958
Validate 6	0.993	0.992	0.971
Validate 7	0.977	0.976	0.942
Validate 8	0.992	0.992	0.942
Validate 9	0.987	0.987	0.897
Validate 10	0.981	0.972	0.777

Table 4.2.2 AUC obtained by ROC after applying 10 rounds of MLP on PITS C dataset

The result came out to be quite high. As we can see from the table above, the values for severity 3 is as high as 0.992 and the least value of AUC is 0.977. Similarly, for severity 4 the values are also very high. The highest value is 0.996, even better than that that obtained for severity 3 and the lowest value is 0.972. There were comparatively lesser number of records in severity 5, but the result obtained from AUC of ROC is good. The highest value is given as 0.971 and the lowest value is given as 0.777. In this way, we can infer the results are quite good and the model is acceptable.

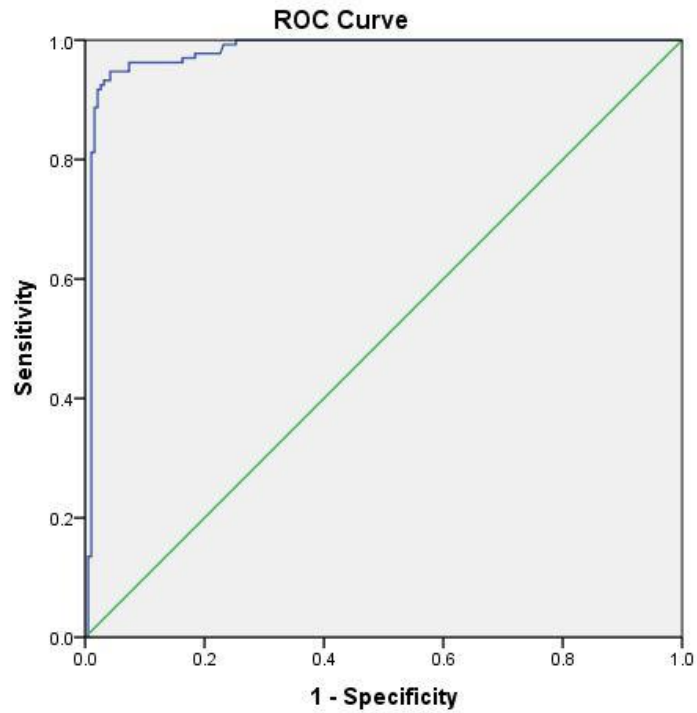


Figure 4.2.5 ROC curve for Severity 3 corresponding to 'validate 1' partitioning variable in PITS C dataset

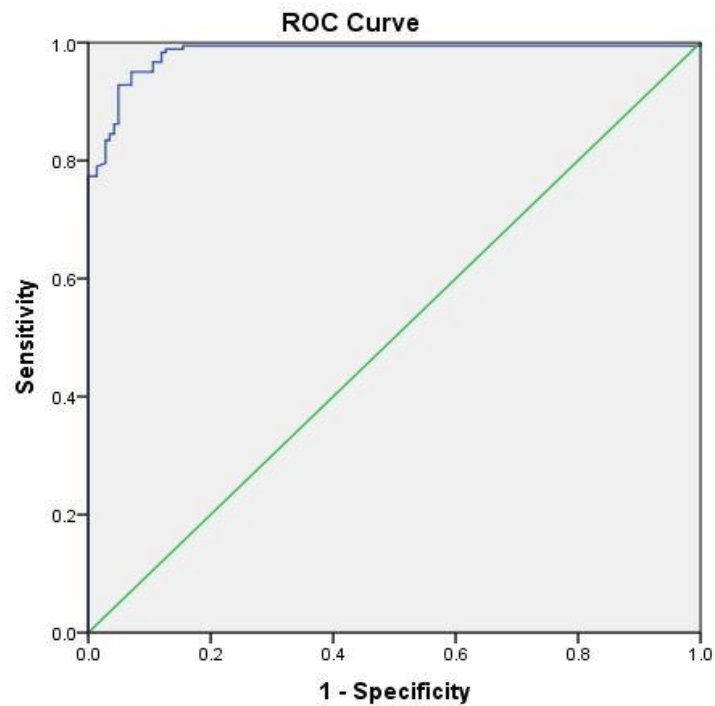


Figure 4.2.6 ROC curve for Severity 4 corresponding to 'validate 1' partitioning variable in PITS C dataset

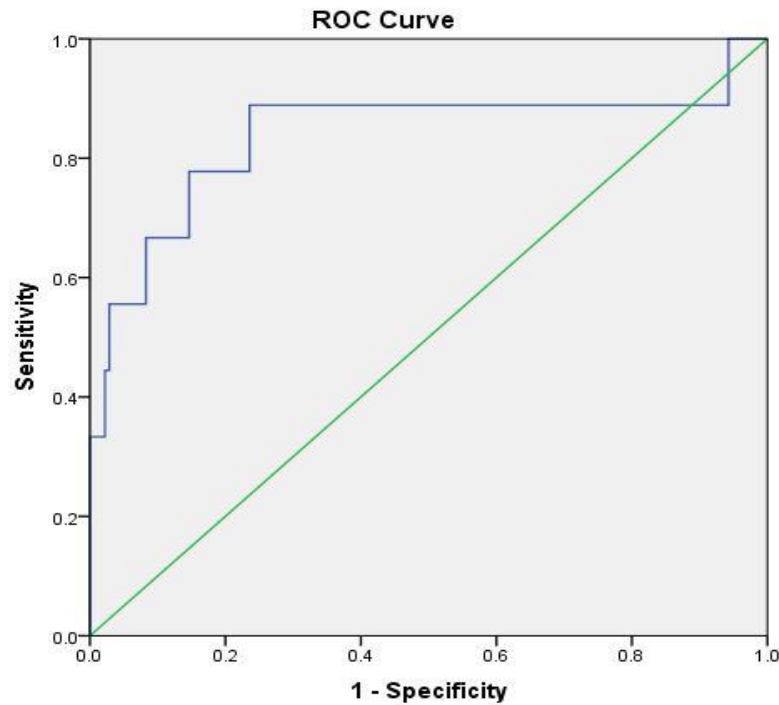


Figure 4.2.7 ROC curve for Severity 5 corresponding to 'validate 1' partitioning variable in PITS C dataset

4.2.6 Multi nominal Logistic Regression

Similar to MLP, we apply multi nominal logistic regression 10 times, once each for a validation parameter. Initially, we select the data by setting the validate parameter value = 1. After the training data is selected we apply the MLR. Here the top 100 words act as the covariates, the severity acts as the dependent. Lastly, we save the estimated response probabilities and predicted category for each variable that is used to analyse the result through ROC curve. MLR is applied only for the training data. Based on the parameters obtained after applying MLR technique, we have manually calculated the values for the test data. Table 4.2.3 displays the results obtained from AUC of ROC technique after applying MLR to the PITS C dataset.

Runs	Severity 3	Severity 4	Severity 5
Validate 1	0.857	0.834	0.623
Validate 2	0.915	0.834	0.574
Validate 3	0.864	0.944	0.957
Validate 4	0.476	0.401	0.723
Validate 5	0.896	0.932	0.683
Validate 6	0.839	0.861	0.446
Validate 7	0.854	0.897	0.896
Validate 8	0.890	0.814	0.651
Validate 9	0.877	0.900	0.571
Validate 10	0.925	0.840	0.783

Table 4.2.3 AUC obtained by ROC after applying 10 rounds of MLR on PITS C dataset

The result came out to be good. As we can see from the table above, the values for severity 3 is as high as 0.925 and the least value of AUC is 0.476. Similarly, for severity 4 the values are also very high. The highest value is 0.944, even better than that that obtained for severity 3 and the lowest value is 0.401. There were comparatively lesser number of records in severity 5, but the result obtained from AUC of ROC is good. The highest value is given as 0.957 and the lowest value is given as 0.571. In this way, we can infer the results are good and the model is acceptable.

The ROC curves obtained for the results of Validate 7 of PITS C dataset using MLR technique are shown below.

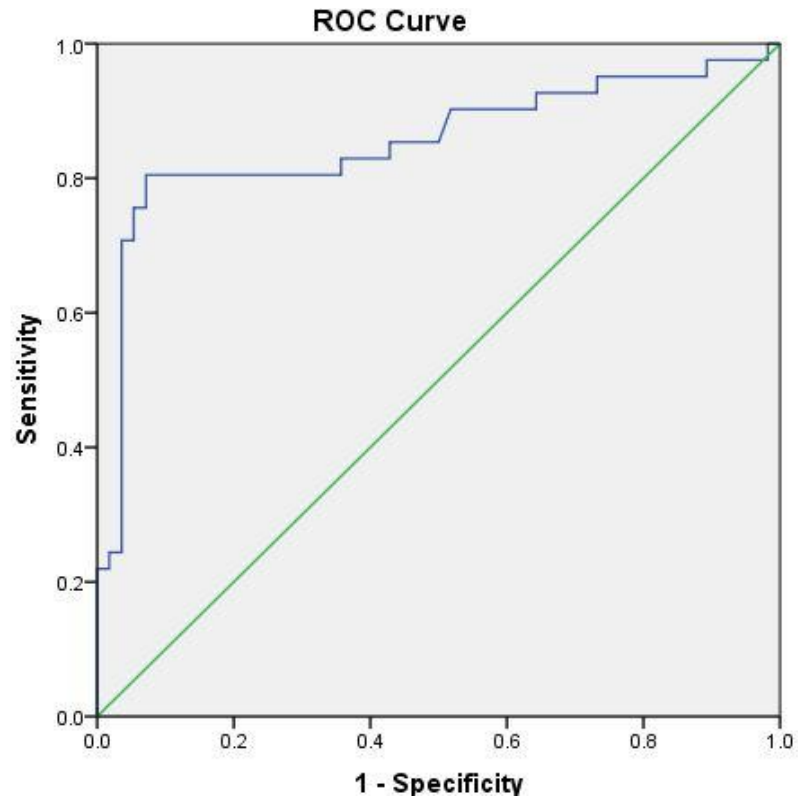


Figure 4.2.8 ROC curve for Severity 3 corresponding to 'validate 7' partitioning variable in PITS C dataset

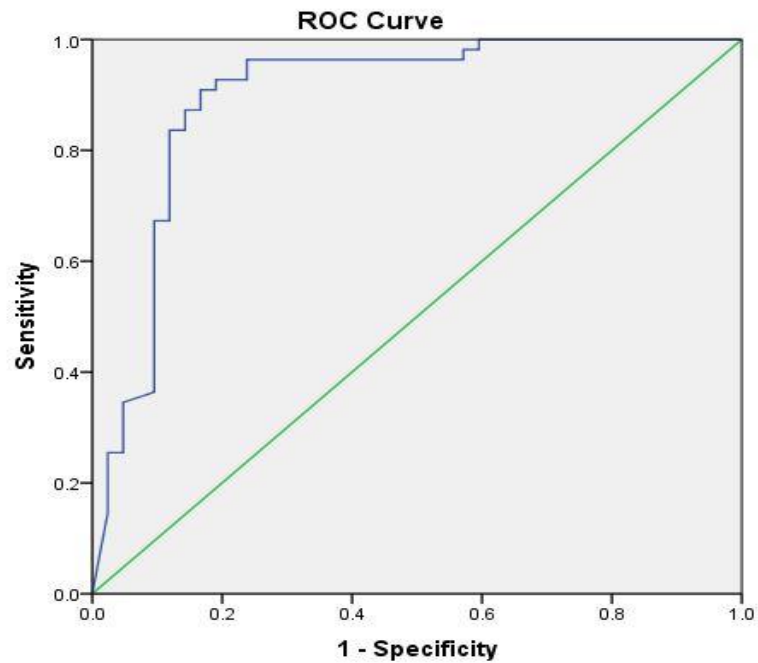


Figure 4.2.9 ROC curve for Severity 4 corresponding to 'validate 7' partitioning variable in PITS C dataset

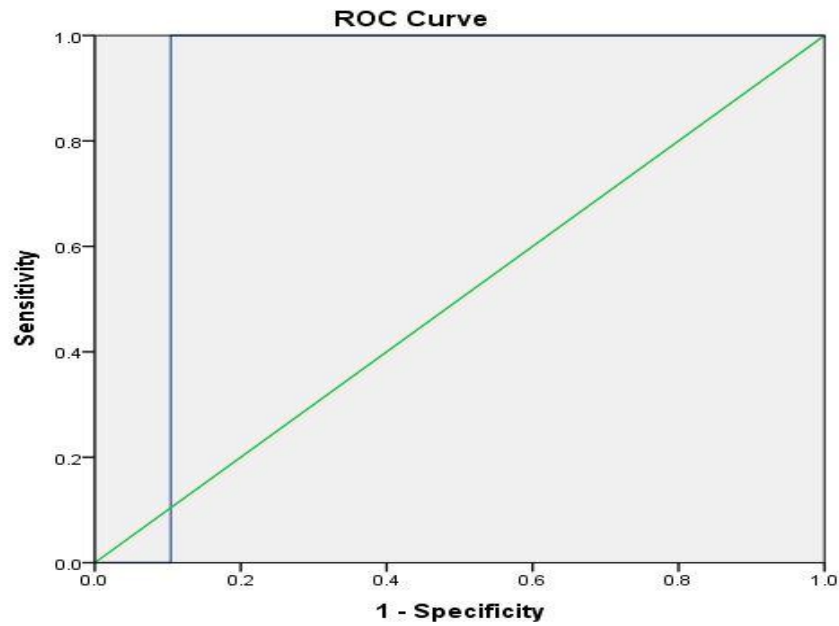


Figure 4.2.10 ROC curve for Severity 5 corresponding to ‘validate 7’ partitioning variable in PITS C dataset

4.2.7 Comparison of Techniques

Table 4.2.5 depicts the output obtained from ROC curve corresponding to two techniques viz. MLR and MLP when applied over PITS C dataset.

Runs	Multi layer Perceptron			Multi nominal Logistic Regression		
	Severity 3	Severity 4	Severity 5	Severity 3	Severity 4	Severity 5
Validate 1	0.980	0.982	0.838	0.857	0.834	0.623
Validate 2	0.991	0.996	0.945	0.915	0.834	0.574
Validate 3	0.989	0.993	0.902	0.864	0.944	0.957
Validate 4	0.988	0.986	0.822	0.476	0.401	0.723
Validate 5	0.989	0.992	0.958	0.896	0.932	0.683
Validate 6	0.993	0.992	0.971	0.839	0.861	0.446
Validate 7	0.977	0.976	0.942	0.854	0.897	0.896
Validate 8	0.992	0.992	0.942	0.890	0.814	0.651
Validate 9	0.987	0.987	0.897	0.877	0.900	0.571
Validate 10	0.981	0.972	0.777	0.925	0.840	0.783

Table 4.2.4 AUC obtained by ROC after applying 10 rounds of MLR and MLP on PITS C data

From, the above table we can easily see that there is quite some difference among the results produced by the two techniques. For severity 3 the highest value obtained by MLP is 0.993 while that of MLR is 0.925 and the corresponding least values are 0.976 and 0.476 respectively. Similarly, for severity 4 the highest value of AUC from MLP is 0.996 while from MLR is 0.932 and the corresponding least values are 0.972 and 0.401 respectively. Severity 5 contains comparatively lesser number of records. The highest value of AUC from MLP for severity 5 is 0.971 while from MLR is 0.957 and the corresponding least values are 0.822 and 0.571 respectively. In this way, we can infer the results from MLP are better in all the cases.

4.3 TOMCAT result

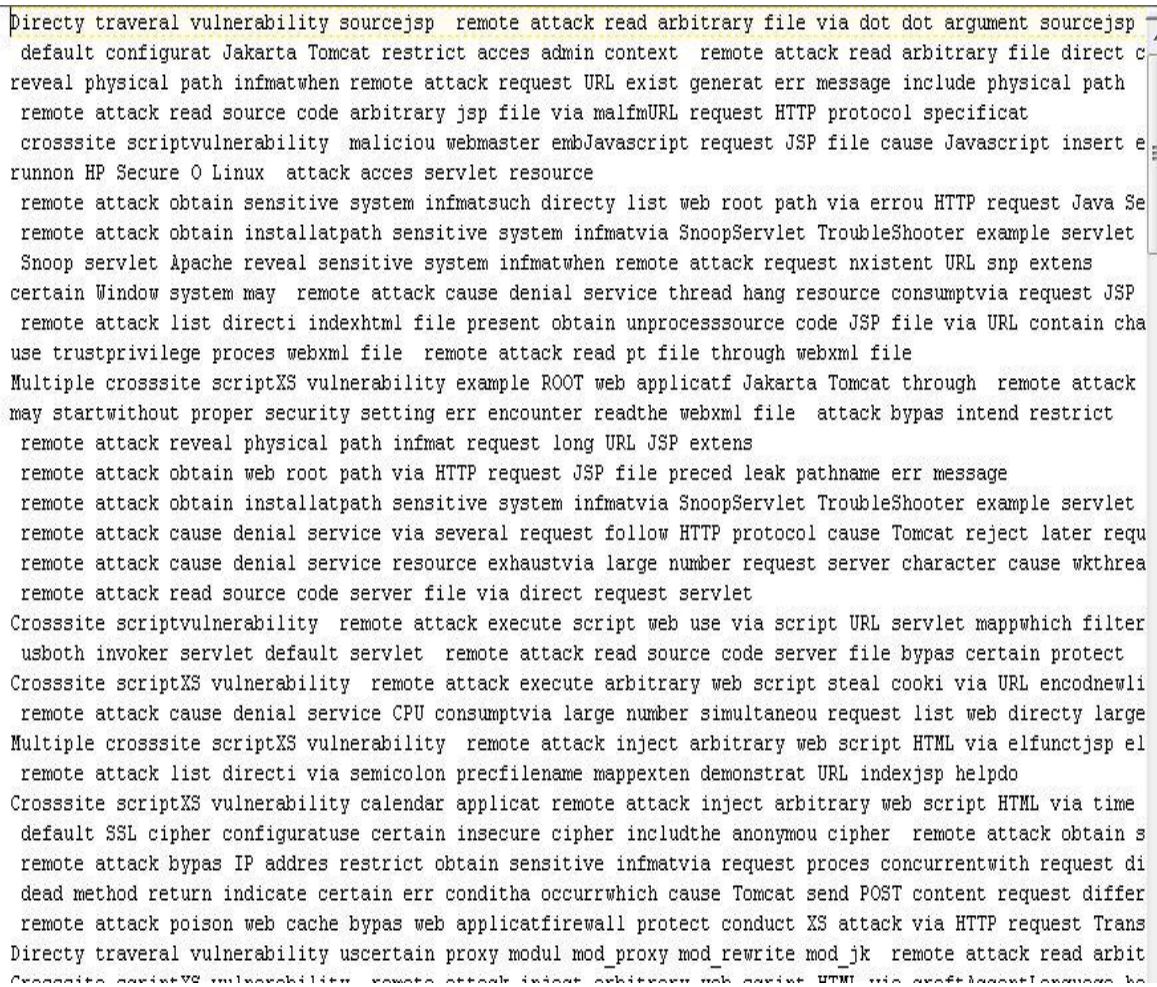
Initially, the defect report data is an XL sheet that contains the complete description of defects. We just choose Description out of it, as this column contains the main information about the defect. A screen shot of the file is given below in figure 4.3.1 for reference.

Defect	Severity	Version
Directory traversal vulnerability in source.jsp al3s remote attackers to read arbitrary files via a .. (dot dot)	1	3
The default configuration of Jakarta Tomcat does not restrict access to the /admin context, which al3s remote attackers can use to read the source code to arbitrary 'jsp' files via a malformed URL request which d	1	3
al3s a remote attacker to read the source code to arbitrary 'jsp' files via a malformed URL request which d	3	3
al3s a remote attacker to read the source code to arbitrary 'jsp' files via a malformed URL request which d	2	3.2.2
A cross-site scripting vulnerability al3s a malicious webmaster to embed Javascript in a request for a .JSP fi	2	3.2.2
running on HP Secure OS for Linux 1.0 al3s attackers to access servlet resources	2	3.2.4
al3s remote attackers to obtain sensitive system information such as directory listings and web root path,	2	3.3.a
al3s remote attackers to obtain the installation path and other sensitive system information via the (1) Sn	3	3.3a
The Snoop servlet under Apache reveals sensitive system information when a remote attacker requests a	3	3.3a
certain Windows systems may al3 remote attackers to cause a denial of service (thread hang and resourc	1	3.3.1
al3s remote attackers to list directories even with an index.html or other file present, or obtain unprocesse	1	3.3.1a
uses trusted privileges when processing the web.xml file, which could al3 remote attackers to read portion:	1	3.3.1a
Multiple cross-site scripting (XSS) vulnerabilities in the (1) examples and (2) ROOT web applications for Jak	2	3.3.2
may be started without proper security settings if errors are encountered while reading the web.xml file, wi	2	4.0.0
al3s remote attackers to reveal physical path information by requesting a long URL with a .JSP extension.	3	4.0.2
al3s remote attackers to obtain the web root path via HTTP requests for JSP files preceded by (1) +/, (2)	3	4.0.2
al3s remote attackers to obtain the installation path and other sensitive system information via the (1) Sn	3	4.1.0
al3s remote attackers to cause a denial of service via several requests that do not fol3 the HTTP protocol,	1	4.1.0
al3s remote attackers to cause a denial of service (resource exhaustion) via a large number of requests to	1	4.1.3
al3s remote attackers to read source code for server files via a direct request to the servlet.	1	4.1.12
Cross-site scripting vulnerability al3s remote attackers to execute script as other web users via script in a l	2	4.1.13

Figure 4.3.1 Initial defect report of TOMCAT data set

4.3.1 Data Pre-processing

Data pre-processing is done using various macros designed in the text pad editor. And then, we achieve a pre-processed data which contains only the root words or the key words that are part of the defect report. After data pre-processing, we get the stemmed text file containing only the key words. This process has to be done with utmost care. Better the stemmed file result, better will be the output after applying machine learning techniques.



```

Directly traversal vulnerability sourcejsp remote attack read arbitrary file via dot dot argument sourcejsp
default configurat Jakarta Tomcat restrict acces admin context remote attack read arbitrary file direct c
reveal physical path infmatwhen remote attack request URL exist generat err message include physical path
remote attack read source code arbitrary jsp file via malfmURL request HTTP protocol specificat
crosssite scriptvulnerability maliciou webmaster embJavascript request JSP file cause Javascript insert e
runnon HP Secure O Linux attack acces servlet resource
remote attack obtain sensitive system infmatsuch directy list web root path via errou HTTP request Java Se
remote attack obtain installatpath sensitive system infmatvia SnoopServlet TroubleShooter example servlet
Snoop servlet Apache reveal sensitive system infmatwhen remote attack request nxistent URL snp extens
certain Window system may remote attack cause denial service thread hang resource consumptvia request JSP
remote attack list directi indexhtml file present obtain unprocesssource code JSP file via URL contain cha
use trustprivilege proces webxml file remote attack read pt file through webxml file
Multiple crosssite scriptXS vulnerability example ROOT web applicatf Jakarta Tomcat through remote attack
may startwithout proper security setting err encounter readthe webxml file attack bypas intend restrict
remote attack reveal physical path infmat request long URL JSP extens
remote attack obtain web root path via HTTP request JSP file preced leak pathname err message
remote attack obtain installatpath sensitive system infmatvia SnoopServlet TroubleShooter example servlet
remote attack cause denial service via several request follow HTTP protocol cause Tomcat reject later requ
remote attack cause denial service resource exhaustvia large number request server character cause wkthrea
remote attack read source code server file via direct request servlet
Crosssite scriptvulnerability remote attack execute script web use via script URL servlet mappwhich filter
usboth invoker servlet default servlet remote attack read source code server file bypas certain protect
Crosssite scriptXS vulnerability remote attack execute arbitrary web script steal cooki via URL encodnewli
remote attack cause denial service CPU consumptvia large number simultaneou request list web directy large
Multiple crosssite scriptXS vulnerability remote attack inject arbitrary web script HTML via elfunctjsp el
remote attack list directi via semicolon precfilename mappexten demonstrat URL indexjsp helpdo
Crosssite scriptXS vulnerability calendar applicat remote attack inject arbitrary web script HTML via time
default SSL cipher configuratuse certain insecure cipher includthe anonymou cipher remote attack obtain s
remote attack bypas IP adres restrict obtain sensitive infmatvia request proces concurrentwith request di
dead method return indicate certain err conditha occurrwhich cause Tomcat send POST content request differ
remote attack poison web cache bypas web applicatfirewall protect conduct XS attack via HTTP request Trans
Directy traversal vulnerability usertain proxy modul mod_proxy mod_rewrite mod_jk remote attack read arbit
Crosssite scriptXS vulnerability remote attack inject arbitrary web script HTML via

```

Figure 4.3.2 Pre processed TOMCAT data set

4.3.2 Dimensionality Reduction

After the stemming process, we list the top 25, 50 and 100 words and focus on all such words for the further course of action. A list of top- 100 words sorted on the basis of their frequency corresponding to TOMCAT dataset is given below. The

classification techniques are yet to be applied on these words.

Attack	Dot	restrict	jsp	Easi
Remote	Obtain	character	large	Enable
Request	Use	connect	sesshijackattack	Filename
Via	Vulnerability	informat	system	Host
Web	Demonstrate	servlet	untrustweb	Invalid
Arbitrary	Traversal	handl	valid	IP
File	Acces	Tomcat	war	Message
Cause	Applicat	contain	body	Modify
Script	Authenticat	multiple	consumptvia	Overwrite
HTML	Intend	proper	craftrequest	Process
Inject	Relatto	URL	entity	Tag
Read	Crosssite	aka	informatsearch	Trigger
Bypass	Err	URI	informatvia	XML
Denial	Sequence	leak	lack	Addres
http	Cooki	nonce	manager	AJP
Parameter	scriptXS	path	name	Attribute
Service	Conduct	server	tld	Check
Header	Value	sessID	applicatto	Chunkheader
Sensitive	Certain	webxml	contextxml	CPU
Directy	Local	ContentLength	data	demonstratvia

Table 4.3.1 Top 100 words of TOMCAT dataset based on the frequency

4.3.3 Generating Tf-Idf matrix

We have used C language to program the above concepts. The file does all the processing and it writes the results in different txt files. The output are two different text files, that contains the infogain values and the Tf*idf values of these words. As per the data, the documents are divided into three severity ratings ranging from 3 to 5, in which the severity rating of 1 means that the defects have a high impact and severity rating of 3 means that the defects have a least impact. We have generated a '100 * 178' matrix of data which contains the Tf*idf values of all the above 100 words document wise. The values are then imported to the IBM SPSS tool for further validation using MLP and MLR technique.

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
D1	0	1	0	1	0	1	1	0	0	0	0	1
D2	1	1	0	0	0	1	1	0	0	0	0	1
D3	1	1	1	0	0	0	0	0	0	0	0	0
D4	1	1	1	1	0	1	1	0	0	0	0	1
D5	0	0	1	0	0	0	1	1	0	0	0	0
D6	1	0	1	0	0	0	0	0	0	0	0	0
D7	1	1	1	1	1	0	0	0	0	0	0	0
D8	1	1	0	0	0	0	0	0	0	0	0	0
D9	1	1	1	0	0	0	0	0	0	0	0	0
D10	1	1	1	0	0	0	0	1	0	0	0	0
D11	1	1	0	1	0	0	2	0	0	0	0	0
D12	1	1	0	0	0	0	4	0	0	0	0	1
D13	1	1	0	0	2	1	0	0	1	3	0	0
D14	1	0	0	0	0	0	1	0	0	0	0	0
D15	1	1	1	0	0	0	0	0	0	0	0	0
D16	1	1	1	1	1	0	1	0	0	0	0	0
D17	1	1	0	0	0	1	0	0	0	0	0	0
D18	1	1	3	1	0	0	0	2	0	0	0	0
D19	1	1	1	0	0	0	0	2	0	0	0	0
D20	1	1	1	1	0	0	1	0	0	0	0	1
D21	1	1	0	1	1	0	0	0	3	0	0	0
D22	1	1	0	0	0	0	1	0	0	0	0	1
D23	1	1	1	1	1	1	1	0	3	0	0	0
D24	1	1	1	0	1	0	2	1	0	0	0	0
D25	1	1	1	1	1	1	0	0	2	1	1	0
D26	1	1	0	1	0	0	0	0	0	0	0	0
D27	1	1	0	1	1	1	0	0	1	1	1	0
D28	1	1	0	0	0	0	0	0	0	0	0	0
D29	1	1	2	0	0	0	0	0	0	0	0	0
D30	0	0	3	0	0	0	0	1	0	0	0	0
D31	2	1	4	1	2	0	0	2	0	0	0	0
D32	1	1	0	1	0	1	1	0	0	0	0	1
D33	1	1	0	1	1	1	0	0	1	1	1	0
D34	0	1	1	1	0	1	1	0	0	0	0	1
D35	1	1	0	0	0	0	0	1	0	0	0	0

Figure 4.3.3 Tf-idf matrix for TOMCAT data set

4.3.4 10-fold cross validation

The Tf-idf matrix is imported to the IBM SPSS tool. We have used Bernoulli's Random function with a probability value of 0.7. We have run this process for 10 times which has created a value of 0 or 1 for each row of dataset in the ratio of 7:3 i.e 70% data is

training data and 30% data is testing data. In this way it helps to run the same data set 10 times for each technique but with the difference that the combination of data in training set or testing set is completely random and different from each other.

attack	remote	request	via	web	arbitrary	file	cause	script	HTML	
.0	.26	.00	1.15	.00	1.67	1.89	.00	.00	.00	
.2	.26	.00	.00	.00	1.67	1.89	.00	.00	.00	
.2	.26	1.31	.00	.00	.00	.00	.00	.00	.00	
.2	.26	1.31	1.15	.00	1.67	1.89	.00	.00	.00	
.0	.00	1.31	.00	.00	.00	1.89	2.02	.00	.00	
.2	.00	1.31	.00	.00	.00	.00	.00	.00	.00	
.2	.26	1.31	1.15	1.72	.00	.00	.00	.00	.00	
.2	.26	.00	.00	.00	.00	.00	.00	.00	.00	
.2	.26	1.31	.00	.00	.00	.00	.00	.00	.00	
.2	.26	1.31	.00	.00	.00	.00	2.02	.00	.00	
.2	.26	.00	1.15	.00	.00	3.78	.00	.00	.00	
.2	.26	.00	.00	.00	.00	7.56	.00	.00	.00	
.2	.26	.00	.00	3.44	1.67	.00	.00	2.35	7.29	
.2	.00	.00	.00	.00	.00	1.89	.00	.00	.00	
.2	.26	1.31	.00	.00	.00	.00	.00	.00	.00	
.2	.26	1.31	1.15	1.72	.00	1.89	.00	.00	.00	
.2	.26	.00	.00	.00	1.67	.00	.00	.00	.00	
.2	.26	3.92	1.15	.00	.00	.00	4.03	.00	.00	
.2	.26	1.31	.00	.00	.00	.00	4.03	.00	.00	
.2	.26	1.31	1.15	.00	.00	1.89	.00	.00	.00	
.2	.26	.00	1.15	1.72	.00	.00	.00	7.04	.00	

Figure 4.3.4 Tf-idf matrix for PITS C data set as imported in SPSS software

val1	val2	val3	val4	val5	val6	val7	val8	val9	val10
1.00	1.00	1.00	.00	1.00	.00	1.00	1.00	1.00	
1.00	1.00	1.00	1.00	1.00	1.00	.00	1.00	1.00	
1.00	1.00	1.00	1.00	.00	1.00	1.00	1.00	1.00	
1.00	1.00	.00	.00	1.00	1.00	.00	1.00	1.00	
1.00	1.00	1.00	.00	1.00	1.00	1.00	1.00	1.00	
1.00	.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
1.00	1.00	1.00	.00	1.00	.00	1.00	1.00	1.00	
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
1.00	1.00	.00	.00	1.00	.00	.00	1.00	1.00	
1.00	1.00	1.00	1.00	1.00	.00	1.00	1.00	1.00	
1.00	1.00	1.00	.00	1.00	.00	.00	.00	.00	
1.00	.00	1.00	1.00	1.00	.00	1.00	1.00	1.00	
1.00	.00	.00	1.00	1.00	1.00	1.00	1.00	1.00	
1.00	.00	1.00	.00	1.00	.00	.00	1.00	1.00	
.00	1.00	1.00	1.00	.00	1.00	1.00	.00	.00	
.00	1.00	.00	1.00	1.00	1.00	1.00	1.00	1.00	
1.00	.00	.00	1.00	1.00	1.00	1.00	.00	1.00	
1.00	1.00	1.00	1.00	1.00	.00	.00	.00	1.00	
.00	.00	.00	.00	.00	.00	1.00	1.00	1.00	
1.00	1.00	.00	.00	1.00	.00	1.00	1.00	1.00	
1.00	1.00	.00	1.00	1.00	1.00	1.00	.00	1.00	

Figure 4.3.5 Tf-idf matrix for TOMCAT data set with 10 validation parameters

4.3.5 Multi layer Perceptron

On the above data we have applied multilayer perceptron (MLP) technique 10 times, once for each validation parameter. Initially, we select the data by setting the validate parameter value = 1. After the training data is selected we apply the MLP. Here the top 100 words act as the covariates, the severity acts as the dependent variable and the validate parameter acts as the partitioning variable. Lastly, we save the predicted value or category for each variable that is used to analyse the result through ROC curve. Table 4.2.2 displays the results obtained after each round of validation for the above mentioned severities.

Runs	Severity 1	Severity 2	Severity 3
Validate 1	0.991	0.922	0.954
Validate 2	0.974	0.851	0.930
Validate 3	0.958	0.831	0.777
Validate 4	0.931	0.724	0.880
Validate 5	0.898	0.797	0.718
Validate 6	0.974	1.0	0.938
Validate 7	0.932	0.661	0.721

Validate 8	0.927	0.686	0.913
Validate 9	0.953	0.823	0.798
Validate 10	0.920	0.859	0.852

Table 4.3.2 AUC obtained by ROC after applying 10 rounds of MLP on TOMCAT dataset

The result came out to be quite high. As we can see from the table above, the values for severity 3 is as high as 0.991 and the least value of AUC is 0.898. Similarly, for severity 4 the values are also very high. The highest value is 1.0, even better than that that obtained for severity 3 and the lowest value is 0.660. There were comparatively lesser number of records in severity 5, but the result obtained from AUC of ROC is good. The highest value is given as 0.954 and the lowest value is given as 0.718. In this way, we can infer the results are quite good and the model is acceptable.

The ROC curves obtained for the results of Validate 4 of TOMCAT dataset using MLR technique are shown below.

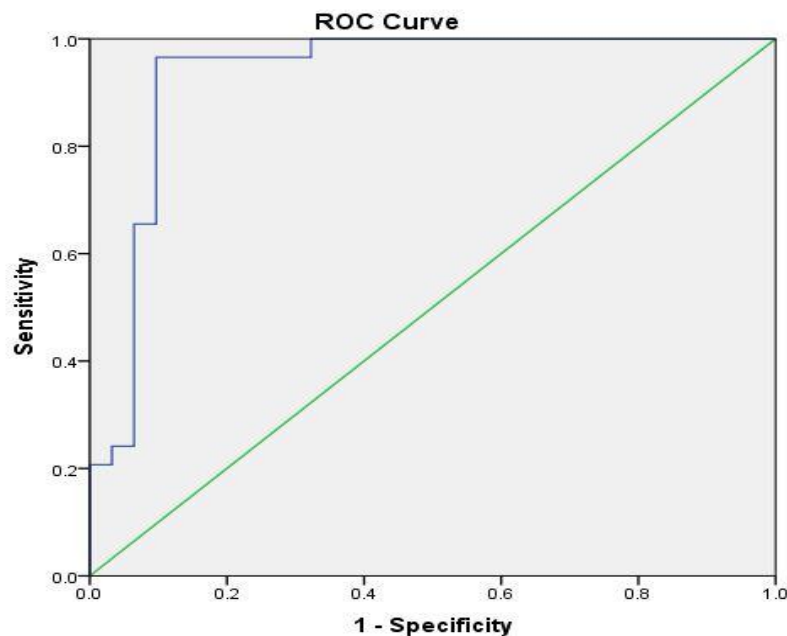


Figure 4.3.6 ROC curve for Severity 1 corresponding to 'validate 4' partitioning variable in TOMCAT dataset

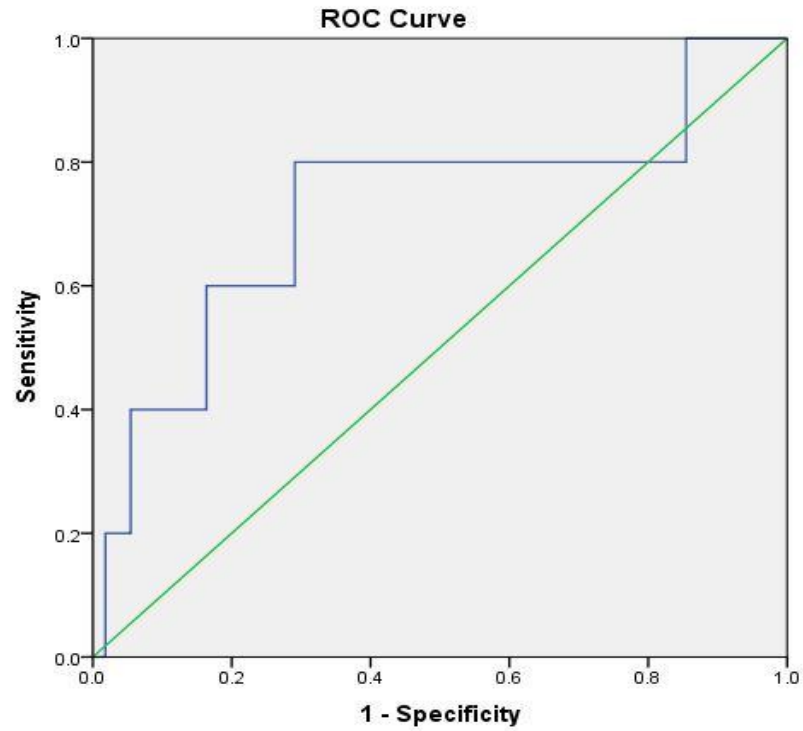


Figure 4.3.7 ROC curve for Severity 2 corresponding to 'validate 4' partitioning variable in TOMCAT dataset

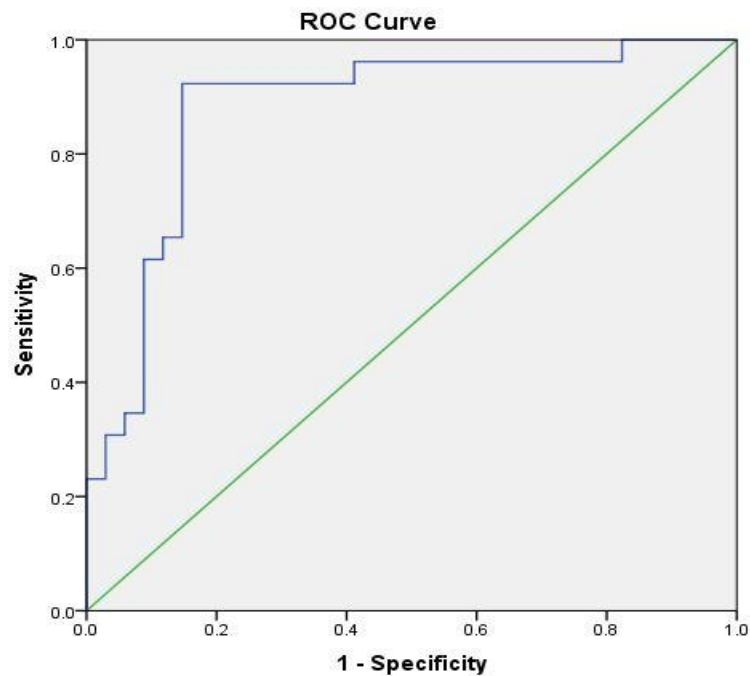


Figure 4.3.8 ROC curve for Severity 3 corresponding to 'validate 4' partitioning variable in TOMCAT dataset

4.3.6 Multi nominal Logistic Regression

Similar to MLP, we apply multi nominal logistic regression 10 times, once each for a validation parameter. Initially, we select the data by setting the validate parameter value = 1. After the training data is selected we apply the MLR. Here the top 100 words act as the covariates, the severity acts as the dependent. Lastly, we save the estimated response probabilities and predicted category for each variable that is used to analyse the result through ROC curve. MLR is applied only for the training data. Based on the parameters obtained after applying MLR technique, we have manually calculated the values for the test data. Table 4.2.3 displays the results obtained from AUC of ROC technique after applying MLR to the TOMCAT dataset.

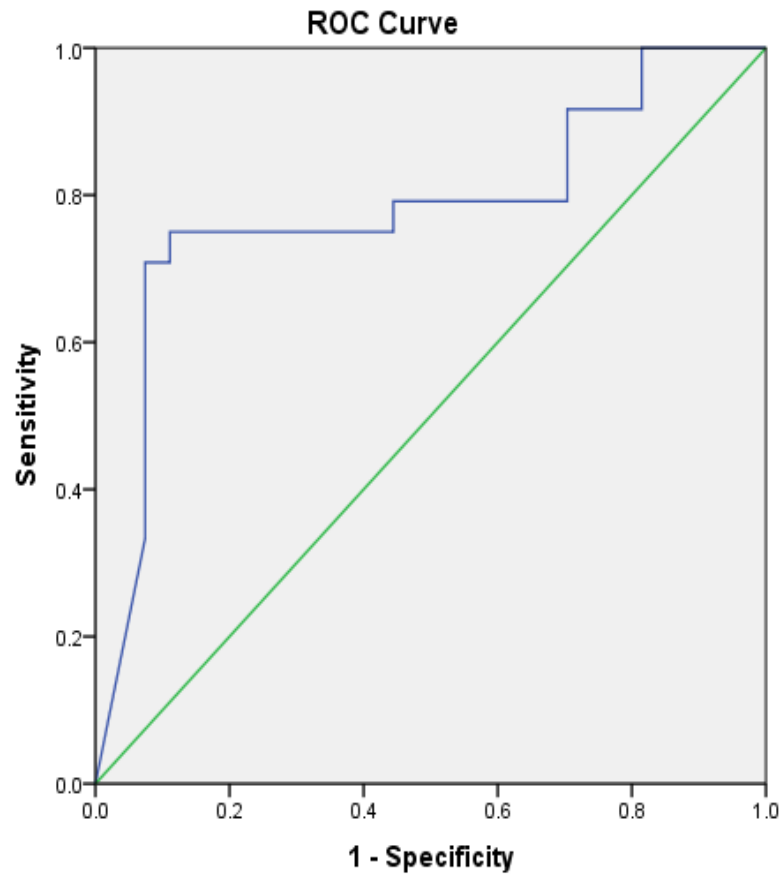
	Severity 3	Severity 4	Severity 5
Validate 1	0.482	0.529	0.513
Validate 2	0.708	0.042	0.570
Validate 3	0.590	0.528	0.603
Validate 4	0.506	0.793	0.650
Validate 5	0.455	0.364	0.641
Validate 6	0.652	0.375	0.617
Validate 7	0.525	0.914	0.503
Validate 8	0.495	0.548	0.559
Validate 9	0.674	0.326	0.594
Validate 10	0.781	0.609	0.752

Table 4.3.3 AUC obtained by ROC after applying 10 rounds of MLR on TOMCAT dataset

The result came out to be average. As we can see from the table above, the values for severity 3 is as high as 0.78 and the least value of AUC is 0.455. Similarly, for severity 4 the values are also very high. The highest value is 0.94, even better than that that obtained for severity 3 and the lowest value is 0.042. There were comparatively lesser number of records in severity 5, but the result obtained from AUC

of ROC is good. The highest value is given as 0.752 and the lowest value is given as 0.503. In this way, we can infer the results are good and the model is acceptable.

The ROC curves obtained for the results of Validate 2 of TOMCAT dataset using MLR technique are shown below.



Diagonal segments are produced by ties.

Figure 4.3.9 ROC curve for Severity 1 corresponding to 'validate 2' partitioning variable in TOMCAT dataset

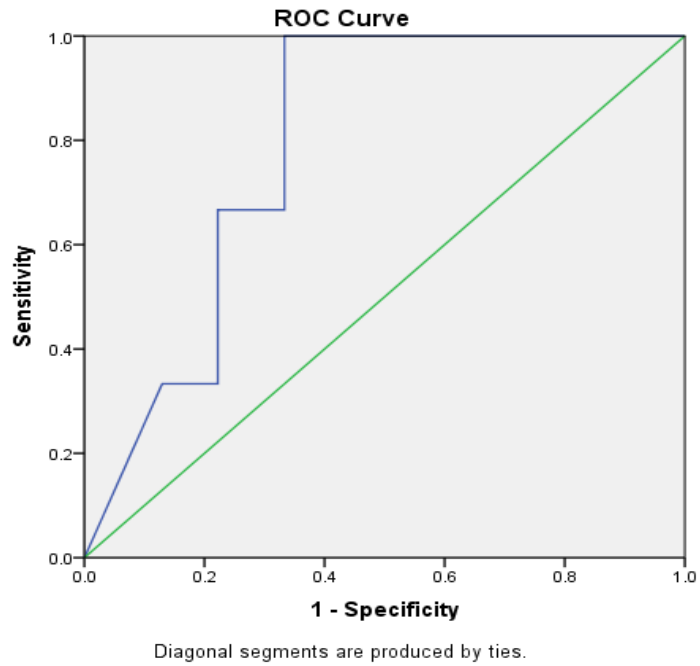


Figure 4.3.10 ROC curve for Severity 2 corresponding to 'validate 2' partitioning variable in TOMCAT dataset

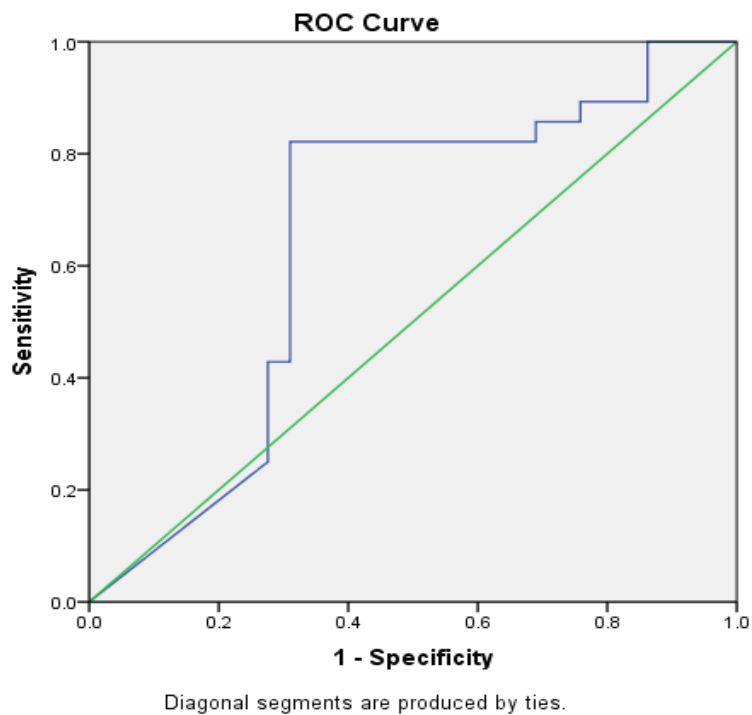


Figure 4.3.11 ROC curve for Severity 3 corresponding to 'validate 2' partitioning variable in TOMCAT dataset

4.3.8 Comparison of Techniques

Of the two techniques that have been used, we see that MLP produces a better output than MLP. Let's see the output obtained from ROC curve for the two techniques when applied over TOMCAT dataset.

	Multi layer Perceptron			Multi nominal Logistic Regression		
	Severity 3	Severity 4	Severity 5	Severity 3	Severity 4	Severity 5
Validate 1	0.991	0.922	0.954	0.482	0.529	0.513
Validate 2	0.974	0.851	0.930	0.708	0.042	0.570
Validate 3	0.958	0.831	0.777	0.590	0.528	0.603
Validate 4	0.931	0.724	0.880	0.506	0.793	0.650
Validate 5	0.898	0.797	0.718	0.455	0.364	0.641
Validate 6	0.974	1.0	0.938	0.652	0.375	0.617
Validate 7	0.932	0.661	0.721	0.525	0.914	0.503
Validate 8	0.927	0.686	0.913	0.495	0.548	0.559
Validate 9	0.953	0.823	0.798	0.674	0.326	0.594
Validate 10	0.920	0.859	0.852	0.781	0.609	0.752

Table 4.3.4 AUC obtained by ROC after applying 10 rounds of MLR and MLP on TOMCAT dataset

From, the above table we can easily see that there is quite some difference among the results produced by the two techniques. For severity 3 the highest value obtained by MLP is 0.991 while that of MLR is 0.781 and the corresponding least values are 0.898 and 0.482 respectively. Similarly, for severity 4 the highest value of AUC from MLP is 1.0 while from MLR is 0.914 and the corresponding least values are 0.661 and 0.042 respectively. Severity 5 contains comparatively lesser number of records. The highest value of AUC from MLP for severity 5 is 0.954 while from MLR is 0.752 and the corresponding least values are 0.718 and 0.513 respectively. In this way, we can infer the results from MLP are better in all the cases.

CONCLUSION

Defect reports are one of the most important outcome of software. They not only give us a track of what all mistakes were done in the project and how they were rectified, but also form a basis for all kinds of similar projects. In any defect report, the description and associated severity gives detailed information about types of issues that took place in the software development and how to tackle if similar issues come again.

Not much work has been done till date in the area of analysing the defect reports in terms of the classification of severity levels. We have analysed the defect reports by applying various text mining techniques in combination with machine learning methods. Text mining technique starts from scanning the defect report, selecting the useful details and then pre-processing them into root words. Then these root words are rated using various parameters such as Info-gain measure and Tf-idf value. The top words so obtained are imported into the IBM SPSS environment in matrix format. Finally, machine learning technique like MLP and statistical technique like MLR are applied on them to assess the severity. We have also used 10-fold cross validation to make sure that the results so obtained are correct and they can easily be referred for future references.

As a case study, we used NASA PITS dataset and TOMCAT dataset. PITS repository contains datasets collected from various scientific and research projects as robotic mission projects over decades. The data consists of description, title and severity of the documents in the project. Similarly, we had collected data from the TOMCAT dataset also that contains defect description, severity and version. We chose the description and severity of data to work upon.

We applied the two techniques MLP and MLR to assess the results. In almost all case, MLP gave better results than MLR. In PITS data set for severity 3 the highest value obtained by MLP is 0.993 while that of MLR is 0.925 and the corresponding least values are 0.976 and 0.476 respectively. Similarly, for severity 4 the highest value of AUC from MLP is 0.996 while from MLR is 0.932 and the corresponding least values

are 0.972 and 0.401 respectively. In TOMCAT dataset for severity 3 the highest value obtained by MLP is 0.991 while that of MLR is 0.781 and the corresponding least values are 0.898 and 0.482 respectively. Similarly, for severity 4 the highest value of AUC from MLP is 1.0 while from MLR is 0.914 and the corresponding least values are 0.661 and 0.042 respectively. Severity 5 contains comparatively lesser number of records.

FUTURE WORK

As our future work, we intend to replicate our work on similar datasets as PITS and TOMCAT in order to get more generalized results. There are a couple of other methods which are available in the literature for stemming and then calculation of infogain and $Tf*idf$ for the generation of matrix. A lot of these methods need to be explored which could produce even more improved result.

Also another aspect which we could work upon is the study of defect reports. Since the defect report is in human readable format, so it calls for still better option of stemming techniques that could improve the file containing the root words. Better the stemming, subsequent techniques which are more of mathematical calculations will give improved results. Not only this, apart from MLP and MLR, there are a lot of other machine learning techniques like Bagging, Boosting, Decision Trees, Bayes Net which can be applied to the dataset which could yield better results. Also, we have used k-fold cross validation method in our project. The same could be tried out by using different validation methods too.

We could look for some technique to generalize the outcomes from one defect reports to other projects in which the severity is not known. Most of the defect reports will have similar kind of words, so using outcome of few data sets we could make some general inferences that if such type of defect occurs or if such root words are found then the severity is high or low. This way it helps in determining severity for different modules of a project beforehand.

REFERENCES

- [1] Tim Menzies, Andrian Marcus, "Automated Severity Assessment of Software Defect Reports", Software Maintenance, 2008. ICSM 2008. IEEE International Conference on 28 Sep 2008 to 04 Oct 2008
- [2] PITS dataset from NASA PROMISE repository, <http://promisedata.org/>
- [3] Anvik, J., Hiew, L., and Murphy, G. C., "Who should fix this bug?" in Proceedings 28th International Conference on Software Engineering (ICSE'06), 2006, pp. 361-370.
- [4] Canfora, G. and Cerulo, L., "How Software Repositories can Help in Resolving a New Change Request", in Proceedings Workshop on Empirical Studies in Reverse Engineering, 2005, pp.
- [5] Canfora, G. and Cerulo, L., "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] Cubranic, D. and Murphy, G. C., "Automatic Bug Triage Using Text Categorization", in Proceedings 6th International Conference on Software Engineering & Knowledge Engineering (SEKE'04), 2004, pp. 92-97.
- [7] Di Lucca, G. A., Di Penta, M., and Gradara, S., "An Approach to Classify Software Maintenance Requests", in Proceedings IEEE International Conference on Software Maintenance, Montréal, Québec, Canada, 2002, pp. 93-102.
- [8] Wang, X., Zhang, L., Xie, T., Anvik, J., and Sun, J., "An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information", in Proceedings 30th International Conference on Software Engineering (ICSE'08), Leipzig, Germany, 10 - 18 May 2008
- [9] Shannon, Claude E. (July–October 1948)., "A Mathematical Theory of

Communication". Bell System Technical Journal **27** (3): 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

[10] Ramos, J. "Using tf-idf to determine word relevance in document queries." Proceedings of the First Instructional Conference on Machine Learning. 2003.

[11] Kwak, C., and A. Clayton-Matthews. "Multinomial logistic regression." *Nursing research* 51.6 (2002): 404.

[12] Malhotra, R., & Jain, A. (2012). Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality. JIPS, 8(2), 241-262.

[12] http://en.wikipedia.org/wiki/Regression_analysis

[13] http://en.wikipedia.org/wiki/Multinomial_logistic_regression

[14] http://en.wikipedia.org/wiki/Multilayer_perceptron

[15] http://en.wikipedia.org/wiki/Artificial_neural_network

[16] <http://en.wikipedia.org/wiki/Tf%E2%80%93idf>

[17] http://en.wikipedia.org/wiki/Information_gain_in_decision_trees

[18] http://en.wikipedia.org/wiki/Information_entropy#cite_note-1

[19] [http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](http://en.wikipedia.org/wiki/Cross-validation_(statistics))