ABSTRACT

Applying NLP techniques to react bug reports provided by the stakeholders as feedback

For the purpose of bugs reports classification, information gathered from the users as feedback or captured from reporting the bugs to the developers, here in this research a tool is developed which are classifying the bugs into their respective bug type with the subsequent classification of the bug reports into their respective components as well.

This study comprises of a bit introduction to the topic followed by review of the related work in the same domain for classification of bug reports. Natural language processing components along with the machine learning techniques are discussed in depth right away. The methods followed by this study is further explained subsequently. The results of this study shows that the tool is quite effective in the classification of bugs reports.

Machine learning techniques are proved quite performing in this study for the classification of bugs. It also showed that the bug reports with the help of machine learning can quickly fixed the bugs by identifying the bug in no time and user experience is also promised. Moreover, the proposed model can be further enhanced in different areas.

KEYWORDS: Natural Language Processing, Machine Learning, Bug Reports, Scikit-Learn, SpaCy, NLTK

Student's Signature

Date

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NLP:	Natural Language Processing
ML:	Machine Learning
LR:	Logistic Regression
SVM:	Support Vector Machine
SVC:	Support Vector Classifier
DT:	Decision Tree
RF:	Random Forest
KNN:	K-Nearest Neighbors
TF-IDF:	Term Frequency, Inverse Document Frequency
NB:	Naïve Bayes
CSV:	Comma-Separated Value
POS:	Part-Of-Speech
NER:	Name Entity Recognition
TP:	True Positive
FP:	False Positive
TN:	True Negative
FN:	False Negative
CM:	Confusion Matrix

1 INTRODUCTION

Nowadays most brands and companies are increasingly adopting a robust software culture that creates the opportunity to have products online and companies can easily read and collect issues about products online easily [1],[2]. In the software product development phase, the purpose of requirements engineering is to identify the requirement of participants, to document and classify requirements, to negotiate needs, to self-regulate and to think about the needs. Requirement Engineering focuses primarily on engaging the needs of the user, developing a plan for the delivery of user needs and getting their feedback [3]. Most of the engineering requirement is concerned with the demands of users, what their needs are. Studies have shown that 90% of release engineers believe that customer feedbacks are very important when testing a system for success or failure [4].

Nowadays, users are looking for more things, in advance that they can ask for, by seeing the same features in another app or sometimes in their search. The success of a software product depends on the acceptance of the users and their feedback plays an important role while the product will determine whether it is successful or failed [3]. That’s why user’s feedback should be taken seriously and modified software according to their requirements.

While providing feedback users provide feedback in the usual way. Like dealing with problems while using software that may resemble system failures, connectivity issues or features were not working as well as they should have been. In these cases, users send a report or report a problem they are using. Occasionally reports or feedback provided by users only contain user information such as how they feel while using an application that does not help the developer team to update the system. Some reports may be incomplete. Overall, feedbacks and reports contain a combination of different types of data.

But in most cases, all reports and feedbacks are kept in the same place and are not ready to take action based on those. In addition to the large amount of important data collected after user interruptions, companies and software developers do not have enough information about how their products are used by users, what features of the system should be changed for a better user experience, what features should be added for future users [5].

Failure to respond to feedback data provided by users may result in software product failure which may also result in the deductions of users. But all feedback provided to users does not apply to software development. That is why information should be sorted and categorized and should be stored in categories. If data is not stored according to developmental stages, it will create more time and less cost. In this way developers can easily take steps such as fixing bugs in the app, analyzing and adding new features requested by users and at the same time can ignore the undesirable ones.

Software bug prediction is in high demand in software development and maintenance phase in the recent years. Types, location, priority, severity, or number based are different aspects which lie under the prediction domain [6]–[9]. The major breakthrough that has been achieved in the bug identification is, reducing the developer time, the maintenance cost and efforts along with the reliability. Different types of statistical models have been developed which relies on the historical data for prediction purpose [10]. The Bug Tracking System (BST) is one of the richest bug information and applications in all information resources [6]. In most of the cases the bugs occur after the development phase and most probably in the testing phase. Bugs are then reported in the BTS as bug reports by the end user or developer in the most of the cases. The impact of the bugs on the users varies hence, the inconvenience is faced. System crash is also one of the scenarios on the worst side while the other have a minor functionality change. So, it's utmost important to deal with the critical bugs at the earliest on top priority. It is of high demand to classify the bug type as many researchers [11]–[13] concluded that majority of the reports submitted were marked as bug but in fact it is not. Hence, the misclassification leads to bias and are having great impact on the prediction performance. With the manual efforts of classifying the bugs, the noise can be reduced but that require lots of efforts in term of time and additional costs. Therefore, a more accurate and automatic system is still in demand to classify a bug for further processing. In paper [13], the researcher investigated the bug classification on auto using text mining techniques but outcome is limited to the description part. In general, the bugs report are semi-structured documents which required a novel approach to deal with.

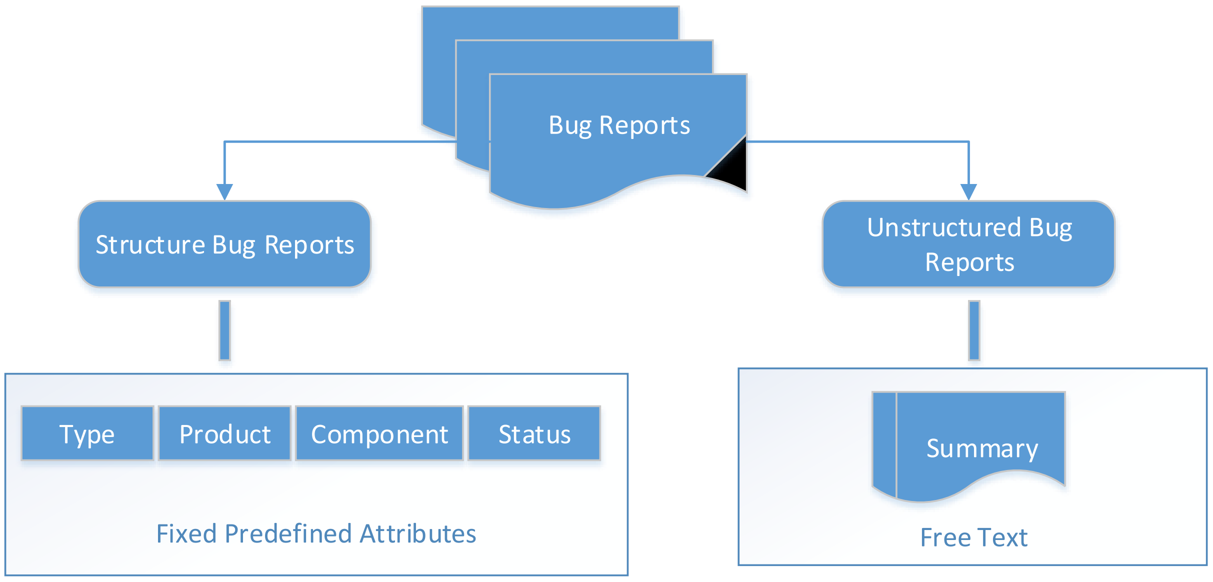


Figure 1: Bug report fields

A few researchers [8], [14]–[16] incorporated multiple fields in the classification but these works are mainly associated with unstructured text and selected fields while few researchers [15] argued that structure text fields play an important factor in the prediction of bug. In this research, main focus is on the type classification with the components as well by merging text mining, natural language processing and machine learning techniques. This research follows a novel method for the classification purpose of bugs.

Below here is the main contribution:

1. A novel approach having multiclass classification of bugs by using Firefox and Thunderbirds datasets.
2. Classified bugs on type as well as on category by processing the summary data.
3. Spacy library is used for preprocessing along with TF-IDF approach and different classifiers are implemented like Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF) and K-Nearest Neighbors (KNN).
4. For evaluation purpose precision, recall, F-value and accuracy are used.

1.1 Background

Here in this section, we mainly focus on background information with respect to software bugs, bug reports, natural language processing along with machine learning, and on the importance of, usage of machine learning and natural language processing for studying software bugs and /or bug reports. A brief description of software bugs and bug reports are also provided which emphasizes the reading importance to control software bugs growing rate. Software bugs and or/ bug reports related research activities with the comparison to this research is also enclosed herein.

In a brief intro a software bug is mainly an error, a failure in a computer program which leads to incorrect results or behave in a strange manner like crashing as well. As the complexity software program increases, the types as well as the number of bugs also increase and software bugs are common in the software development industry. From the submitted papers in software bug/ bug report it clearly indicate that many efforts are already put to classify bug reports, pointing out and correcting the bugs [17]

Software bug reports are mainly clear documents which contain an error logs, bug replication steps, version, product, platform and information regarding the application. For tacking easiness of the progress bug can be written as confirmed, unconfirmed, new, duplicated, closed, fixed etc. A quality bug report should have clearly all the details and specifying the bug properly with clear info to clarify the problem. With contrast to the effectiveness and clarity of the bug report, bug reporters are having no boundness in the writing pattern like writing in a free manner having external information and including irrelevant information. So, bottom line is that a tool is of high demand and helpful to extract the useful information with discarding the irrelevant for such type of bug reports.

1.2 Scope of the work

The growing data in software programs has provided a unique opportunity for all software development organizations to use it in decision making. A variety of data such as bug reports, website interaction details, product usage ratings, or test results from software firmware companies appear and appear as structured data that is not properly structured. The data is very scattered and not in a systematic way for further use. The data, when analyzed in detail, are useful for many purposes, especially decision making. Decisions can be at the business level or at the production level. Here in this research, we have categorized different types of data collected by organizations such as Firefox and Thunderbirds, through literature review, research study, natural language processing and quality analysis. Based on the classification we usually make different types of decisions that are made during the software development process cycle. Combining the two we create a model to classify a bug on the type and subsequently classify that on the components as well. This model is a tool to assist a professional who wants a clear understanding of what kind of data can be used for any kind of decisions. Outlines on how to do more research in this area are also highlighted.

1.3 Outline

This thesis is organized as shown in Fig. 1

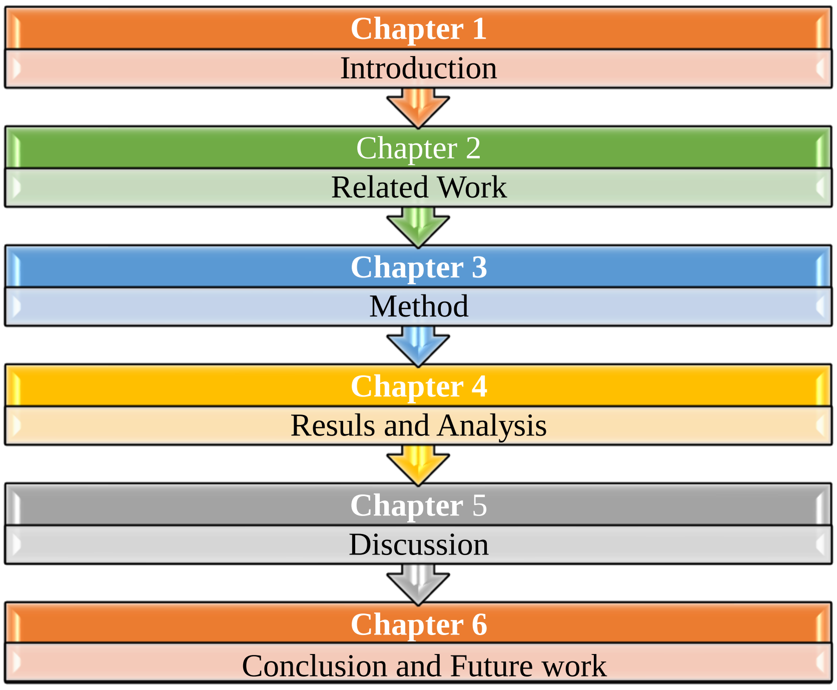


Figure 2 Thesis Outline

Chapter 1 of the thesis describes the introduction to data driven with the background and scope of this work. **Chapter 2** includes related work. **Chapter 3** covers the theoretical background of the techniques used in proposed method. **Chapter 4** includes the implementation of techniques/algorithms subsequently and its analysis, **Chapter 5** convey the discussion in many perspectives, while **Chapter 6** concludes this thesis with future work recommendations.

2 RELATED WORK

Natural language processing, play a vital role in the text classification and have multiple applications in the domain like sentiment analysis, spam detection etc. As the text data are unstructured and extracting the information from the unstructured data is quite tedious and expensive task in term of time with cost.

Many researchers have worked on dealing with the bug reports generated from the bug tracking repository to classify the bugs into different categories to reach on them at the earliest. Research done by Dikshah Behl, Sahil Handa and Anuia Arora, implemented the use of TF-IDF with Naïve Bayes classifier [18] to do the classification of bugs into security and no-security categories. A vocabulary of bugs was gathered from the bug reports and converted each bug report into a vector which indicate mainly the occurrence of certain words in the bug report. On the basis of the weighted values of each complete vector, bug report was classified to either security or non-security bug.

The researcher also worked like in [19] on classification of bug report to either a valid bug or not a valid bug means invalid bug. In this system N-gram Inverse Document Frequency is used for the derivation of phrases from the bug report. The feature extracted are then poured as input to the two models – Logistic regression and Random Forest classification. The approach followed by these researchers was compared with the topic modeling approach for duplicate bug classification. In the research it concluded that the Ngram IDF was having high accuracy in both models.

Toa Zhang and Byungieong Lee, published a paper [20], in which they discussed a method for identical bug detection reported by Bugzilla. Identification of duplicates was found by the use of bug rules and text-based similarities. The appropriate category was classified by using the taxonomy classification and hierarchical clustering algorithm.

The author Neelofar and co-authors [21], compared feature extraction methods both TF-IDF and Chi Squre algorithm for the classification of bugs in their research. The author concluded that the Chi Squared algorithm performed well than the TF-IDF in achieving better accuracy. The main idea for the classification was to correctly classify the bugs which help the bug triage team in the assignment of the reported bug to the right developer for further processing.

Lin Tan and co-authors, proposed an approach in their research [22], of analyzing the bug report and created on the root cause a patch-based approach. R2Fix then further classifies the bug into respective categories, which obtains the parameters that are used in the source code based on the bug report from the classifier and for the generation a patch uses these patterns.

Aishwarya and co-authors, implemented SVM, DT, RF and Naïve Bayes for the bug classification and used two level of classification in their research paper [23].

Bug triaging helps to predict whether the bug report is a real bug or not. The severity classification process is tedious and time consuming due to the large number of bug reports in the BTS. For a long time, researchers have been trying to perform a bug triage procedure. Decent success has been accomplished in the same way. The list of such activities is fully explored in this section as follows: In a previous study, researchers classified the binary bug severity into Sever and Non-Sever / Bug categories and non-bug categories.

2.1 Bug class and non-bug class

Antoniol et al. published in research paper [13], an automatic bug classification of classifying the bug into a valid or invalid bug. In this research the researcher extracted the feature by Acrive Directory Tree (ADT) and vector space technique. Machine learning algorithms like Naive Bayes (NB) and Linear Logistic regression (LLR) are used for the training purpose. The results of this model show the precision of Eclipse, JBoss and Mozilla project with the range of 77% to 82%. Pingclasai et al. [24], also worked in the same area for the classification of bug in to bugs or non-bugs. They used LDA with LLR and NB for the classification purpose. The precious varied for HTTPClient, Lucene, Jackrabbit from 66% to 76%, 71% to 82% and 65% to 77% respectively. Zhou et al. [16] combine both text and data mining algorithms. With text mining algorithm, summary of bug reports is extracted and input to the machine leaner with the other structured input. For the combination of the stages data drafting algorithms are used. The execution of this model showed quite promising results by increasing the f-measure. The range of f-measure varies for JBoss, Mozilla and Eclipse like 84% to 93%, 78% to 81% and 72% to 79% respectively. Chawla [25] used latent semantic indexing (LSI) for severity classification with TF-IDF feature extracted techniques. For better results Fuzzy logic algorithm is used for the training purpose.

2.2 Critical Classes and Non-Critical Classes

Lamkanfi, the author described the NB classifier with the collaboration of TF-IDF for the classification of bugs either in one of the categories that is sever and non-sever. Overall 65 to 85% accuracy is achieved from that of Gnome, Eclipse and Mozilla. The author Saihaan [26], also did the similar work and have achieved high accuracy of 99.83%. Nagwaniet, also worked on the classification of bug report as bug or non-bug. In this research the author used simple Random Sampling (RSS) algorithm for the taxonomic terms to generate with and evaluated the accuracy of JBoss-Seam, Android, Mozilla and MySQL. Kantiet [27], utilized the chi-Squared feature extraction methods with the usage of bigram. Here the author concluded that the performance is improving with the usage of n-gram application. Gujral, describe in the research paper [28] and utilized TF-IDF feature extraction with the combinition of Naive Bayes Multinomial (NBM) classifier. With this selection the author obtained the accuracy of 69% and precision of 70% with Eclipse bug reports. The author Sharma used (In-foGain and Chi-square) as two feature selection techniques. The vocabulary of critical words was created by selected terms and K-NN and NBM was used for the training the model. The author described that K-NN classifier is having better accuracy with the range of 69% to 75%. Pandey concluded in research paper [29] with the different classifiers implementations like K-NN, NB, RF, SVM and LDA for the classification of the bugs as sever and non-sever class and achieved the accuracy of 75% to 83%. Gegicket, the author and co-authors also worked on the bug severity classification with the use of SAS text miner with SVD on the dataset of Cisco and the finding was that the misclassified bug reports were of 77%. They compared Chi-Square, In-foGain and correlation coefficient with NBM classifier and was concluded that the feature selection plays a major role in the precision of the dataset. The author Jin and the co-authors, in their findings showed a better result on Lamkanfi data set for the normal severity reports.

For the above discussion it is clear that there must of an optimistic bug fixing due to the effect of harmful impact on the software. As the classification task for bug classification is manual although some researchers tried to automate the process with the help of machine learning but still there is a room for improvement and reliability of the models which must be taken in a professional way to predict the bugs, it's type and the component in which the bug relate to.

3 METHOD

In this section, the proposed approach is discussed in depth. The overall framework of this research is given herein:

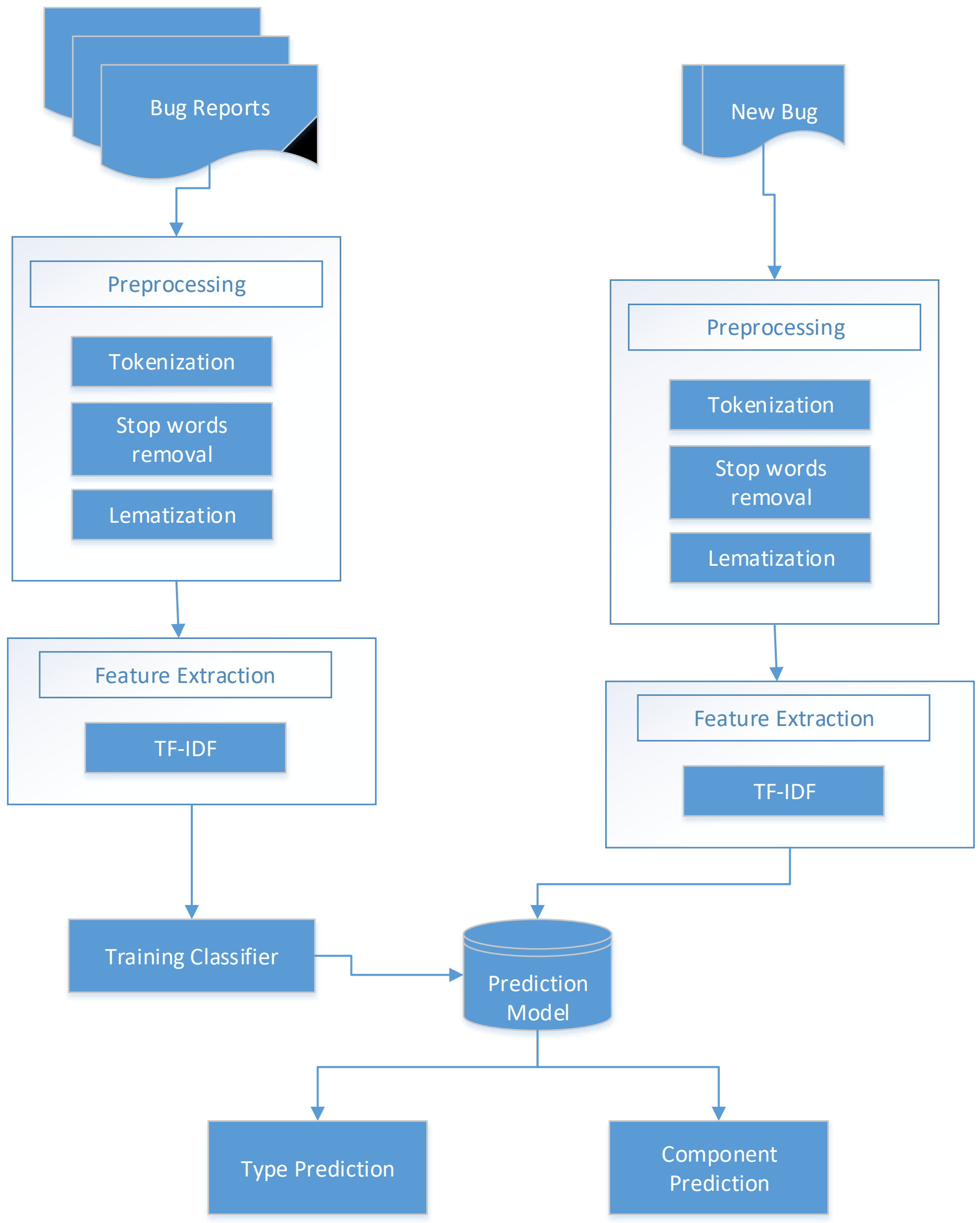


Figure 3 Proposed Methodology

Detailed methodology is given in fig. 3 above and the steps are discussed below.

3.1 Data Acquisition

The experimental datasets for this research are chosen Firefox and ThunderBird which are available publicly for research purposes.

3.1.1 Firefox

Firefox dataset is available in comma-separated value (CSV) format and the data of one entry is shown in Table 1.

Table 1 Firefox bug features

Bug ID	**1695535**
Type	defect
Summary	Could not write session state file Error: TypeError: cannot use 'in' operator to search for "toMsg" in "out of memory"
Product	Firefox
Component	Session Restore
Assignee	nobody
Status	New
Resolution	-----
Updated	2/28/2021 14:28

The dataset of Firefox that are used for this research contain total of 10 thousand instances with nine features as shown in Table 1. The bug Id is the unique Id for each instance, so here we are having total of 10 thousand unique Bug Id's. The type indicates the type of bug, we are having three types of bugs like defect, enhancement and task. Product is straight forward which is clearly Firefox for all the instances of this dataset. The component is where the bug locates to, all the components are shown in Table 2.

Table 2 Firefox Components

Normandy Server	Address Bar	Downloads Panel	Tours	Pioneer
Session Restore	New Tab Page	PDF Viewer	Normandy Client	WebPayments UI
Theme	Sync	about: logins	Enterprise Policies	Firefox Monitor
Firefox Accounts	File Handling	Security	Migration	Remote Settings Client
Bookmarks & History	Site Permissions	Site Identity	Shell Integration	Distributions
Toolbars and Customization	Pocket	Top Sites	Protections UI	Translation
General	Installer	Keyboard Navigation	System Add-ons: Off-train Deployment	Extension Compatibility
Tabbed Browser	Messaging System	Preferences	Page Info Window	
Search	Nimbus Desktop Client	Screenshots	Disability Access	
Menus	Headless	Private Browsing	Launcher Process	

Assignee is whom the bug assigned, status is either new or unconfirmed while the resolution and updated are straight forward and updated feature shows the updated date of the bug.

3.1.2 ThunderBird

Same as Firefox, ThunderBird dataset is available in comma-separated value (CSV) format and the data of first entry is shown in Table 3.

Table 3 ThunderBird bug features

Bug ID	**1559448**
Type	defect
Summary	Shutdown crash [@ AsyncShutdownTimeout | profile-change-teardown | Extension shutdown: wetransfer@extensions.thunderbird.net]
Product	Thunderbird
Component	FileLink
Assignee	goeff
Status	Reopened
Resolution	-----
Updated	2/28/2021 14:28

The dataset of Thunderbird that are used for this research contain total of 7867 instances with nine features as shown in Table 3. The bug Id is the unique Id for each instance, so here we are having total of 7867 unique Bug Id's. The type indicates the type of bug, we are having three types of bugs like defect, enhancement and task. Product is straight forward which is clearly Thunderbird for all the instances of this dataset. The component is where the bug locates to, all the components are shown in Table 4.

Table 4 Thunderbird Components

Message Compose Window	Message Reader UI	Mail Window Front End	Instant Messaging
FileLink	Add-Ons: General	Add-Ons: Extensions API	Search
Untriaged	Filters	Address Book	Testing Infrastructure
OS Integration	Security	Theme	Help Documentation
Folder and Message Lists	Preferences	Disability Access	Migration
General	Toolbars and Tabs	Build Config	
Installer	Account Manager	Upstream Synchronization	

Assignee is whom the bug assigned, status is either new or unconfirmed while the resolution and updated are straight forward and updated feature shows the updated date of the bug.

3.2 Preprocessing

In the preprocessing phase the data is cleaned by applying different techniques. First of all, punctuation is removed from the summary feature with the help of built-in string punctuations like comma, period, hyphen, quotes etc. All the punctions that were removed are:

!"#$%&′() ∗ +, −./: ; <=>? @[\]^_`{|}~

On the summary text, standards steps like Tokenization, stop-words removal and Lemmatization are applied. For this the purpose of preprocessing Spacy library is used. As a whole all these steps lie under the broad term Natural Language Processing (NLP)

3.2.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) [30] is the field of computer science which is basically evolved from artificial intelligence. Natural language processing enable computer to understand human languages. With the combination of Natural language processing and machine learning major break throughs are achieved like computers can predict on human language inputs. Here NLP play a building block in this process of classification of bugs. NLP perform the foundation for the machine leaning model. A proper and dedicated approach should be followed in this phase for generating better results from the machine learning model.

3.2.2 SpaCy

SpacCy is open-source python library which is mainly used for high level Natural Language Processing (NLP). SpaCy library has many features like it support 64 languages, linguistically motivated tokenization and have high level components for part-of-speech tagging, entity named recognition, dependency parsing, sentence segmentation along with lemmatization.

3.2.3 Tokenization

Tokenization is the splitting of text data into token as shown in the figure 4 below. In tokenization the text is splitting on whitespace, prefix, suffix and are one some exceptions as well. Tokenization work like a paragraph is first splitting into sentences and then the sentences are splitting into words. Tokenization is the important step in natural language processing as computer can not understand the textual data and only perform or understand better on the numerical data. So, for the purpose of machine learning words are used as features and hence the tokenization is performed on the textual data.

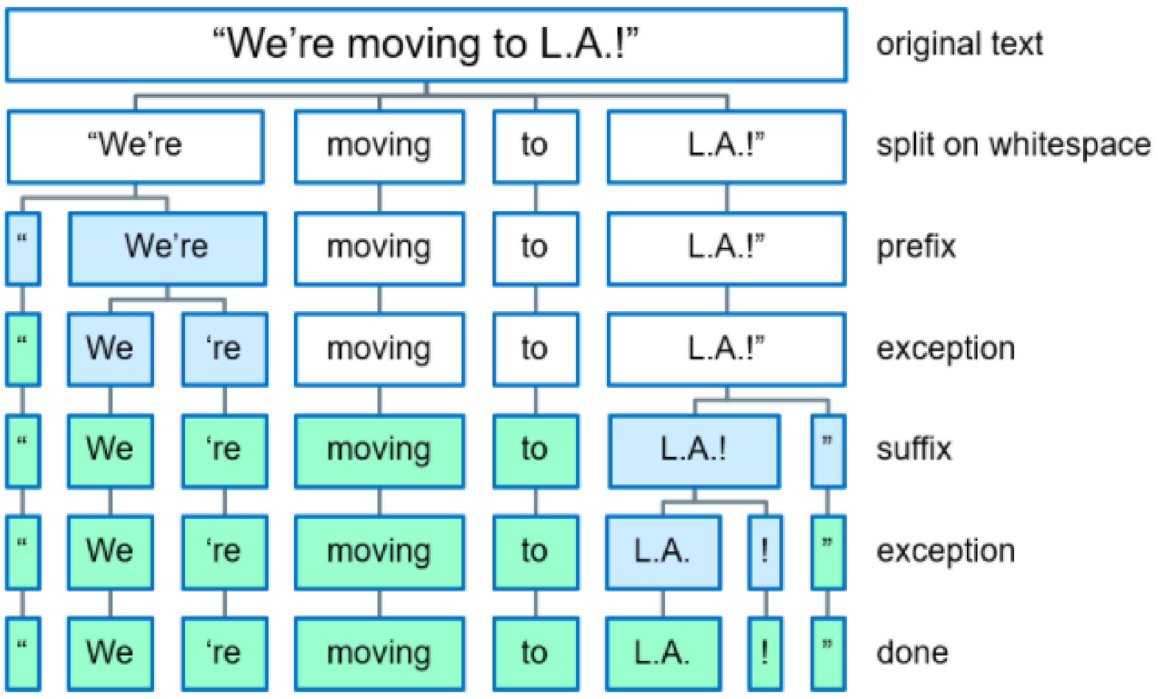
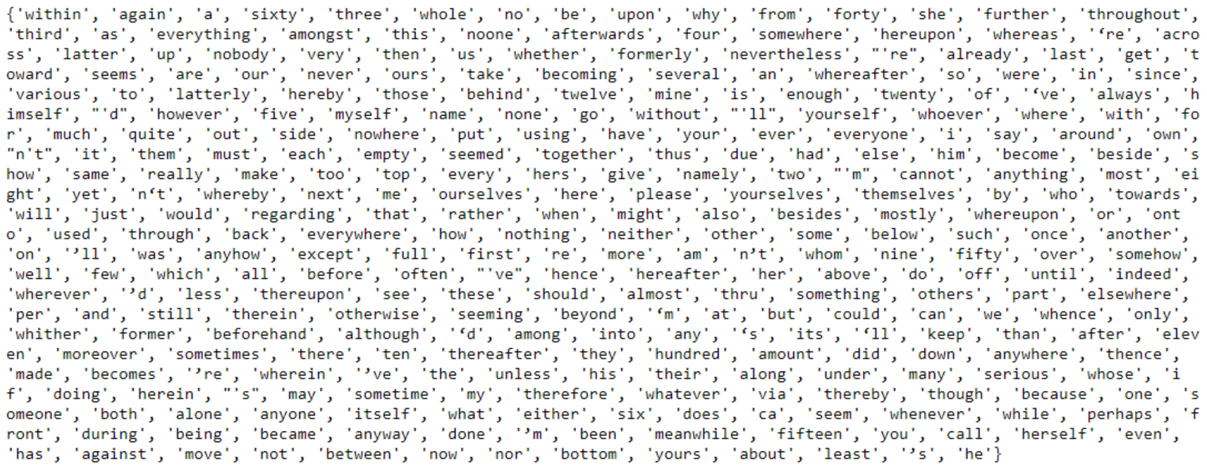


Figure 4 Tokenization

3.2.4 Stop words removal

With the removing unnecessary words from the features will lead the machine learning model to high accurate model as we need such words which are playing key role in the prediction and remove the unwanted words. Such words are called stop words and should be removed before stepping a head to the next step. Here a total of 326 stop words that are removed from the text data. The stop words are shown in fig. 5.



```
{'within', 'again', 'a', 'sixty', 'three', 'whole', 'no', 'be', 'upon', 'why', 'from', 'forty', 'she', 'further', 'throughout',
'third', 'as', 'everything', 'amongst', 'this', 'noone', 'afterwards', 'four', 'somewhere', 'hereupon', 'whereas', '‘re', 'acro
ss', 'latter', 'up', 'nobody', 'very', 'then', 'us', 'whether', 'formerly', 'nevertheless', "'re", 'already', 'last', 'get', 't
oward', 'seems', 'are', 'our', 'never', 'ours', 'take', 'becoming', 'several', 'an', 'whereafter', 'so', 'were', 'in', 'since',
'various', 'to', 'latterly', 'hereby', 'those', 'behind', 'twelve', 'mine', 'is', 'enough', 'twenty', 'of', '‘ve', 'always', 'h
imself', "'d", 'however', 'five', 'myself', 'name', 'none', 'go', 'without', "'ll", 'yourself', 'whoever', 'where', 'with', 'fo
r', 'much', 'quite', 'out', 'side', 'nowhere', 'put', 'using', 'have', 'your', 'ever', 'everyone', 'i', 'say', 'around', 'own',
"n't", 'it', 'them', 'must', 'each', 'empty', 'seemed', 'together', 'thus', 'due', 'had', 'else', 'him', 'become', 'beside', 's
how', 'same', 'really', 'make', 'too', 'top', 'every', 'hers', 'give', 'namely', 'two', "'m", 'cannot', 'anything', 'most', 'ei
ght', 'yet', 'n‘t', 'whereby', 'next', 'me', 'ourselves', 'here', 'please', 'yourselves', 'themselves', 'by', 'who', 'towards',
'will', 'just', 'would', 'regarding', 'that', 'rather', 'when', 'might', 'also', 'besides', 'mostly', 'whereupon', 'or', 'ont
o', 'used', 'through', 'back', 'everywhere', 'how', 'nothing', 'neither', 'other', 'some', 'below', 'such', 'once', 'another',
'on', '’ll', 'was', 'anyhow', 'except', 'full', 'first', 're', 'more', 'am', 'n’t', 'whom', 'nine', 'fifty', 'over', 'somehow',
'well', 'few', 'which', 'all', 'before', 'often', "'ve", 'hence', 'hereafter', 'her', 'above', 'do', 'off', 'until', 'indeed',
'wherever', '’d', 'less', 'thereupon', 'see', 'these', 'should', 'almost', 'thru', 'something', 'others', 'part', 'elsewhere',
'per', 'and', 'still', 'therein', 'otherwise', 'seeming', 'beyond', '‘m', 'at', 'but', 'could', 'can', 'we', 'whence', 'only',
'whither', 'former', 'beforehand', 'although', '‘d', 'among', 'into', 'any', '‘s', 'its', '‘ll', 'keep', 'than', 'after', 'elev
en', 'moreover', 'sometimes', 'there', 'ten', 'thereafter', 'they', 'hundred', 'amount', 'did', 'down', 'anywhere', 'thence',
'made', 'becomes', '’re', 'wherein', '’ve', 'the', 'unless', 'his', 'their', 'along', 'under', 'many', 'serious', 'whose', 'i
f', 'doing', 'herein', "'s", 'may', 'sometime', 'my', 'therefore', 'whatever', 'via', 'thereby', 'though', 'because', 'one', 's
omeone', 'both', 'alone', 'anyone', 'itself', 'what', 'either', 'six', 'does', 'ca', 'seem', 'whenever', 'while', 'perhaps', 'f
ront', 'during', 'being', 'became', 'anyway', 'done', '’m', 'been', 'meanwhile', 'fifteen', 'you', 'call', 'herself', 'even',
'has', 'against', 'move', 'not', 'between', 'now', 'nor', 'bottom', 'yours', 'about', 'least', '’s', 'he'}
```

Figure 5 Stop Words

3.2.5 Lemmatization

For further reducing of dimensionality of the features, either stemming or lemmatization is implemented for the betterment. Stemming is basically removing the end of the words to somehow replace the word with the root word. As stemming is not perfect as it's like implementing a straight forward approach which is not a goof fit for all the words. On the other side lemmatization work on the words tags and replace the words with the correct root word. So, transformation of the words with lemmatization is accurate and we performed lemmatization instead of stemming.

3.3 Feature extraction

In text mining, feature extraction is basically the extraction of words as feature from the textual data. The words are to be converted into numerical data to process with machine learning. So, the words need to be converted into vectors for better processing. After lemmatization, conversion of words to vector is required. So, we process the textual feature from TFIDFVectorizer which convert the feature to the matrix of TF-IDF features. TF-IDF is the abbreviation of term frequency, inverse documents frequency. TF-IDF further reduce the dimensions. Further tfidf score is assigned which help in further processing. For the purpose of training the model we'll only deal with summary feature, type of bug and the component as the rest of the feature don't play any vital role in the output accuracy. So, data is further fed in to the machine leaning model for training purpose.

3.4 Machine learning

Machine learning [31] is basically the study/science to program computers, that they can learn from data and make predictions. Machine learning is currently the most adoptable tool for current word and many researches are ongoing in the field of machine learning ranging from health to business, from stock market to COVID-19 predictions. As discussed in section 2 many researchers worked on the bug classification using different machine learning algorithms. Below here we'll discuss some of the algorithms that we have implemented for the current research.

3.4.1 Classifiers in Machine Learning

There are three major type of machine learning:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

In supervised learning, we input data with with labels while in unsupervised we don't have labels with inputs. Labels are targets for the input. As the name indicated, this type is supervised by the labels while the reinforcement work on the reward algorithms.

Bug classification in this research is a supervised learning as we input the data with the labels for the training purpose and in the testing phase, we don’t input the labels or the targets and the system predict the target which is the beauty of machine learning. There are many classifiers in machine learning but the implemented one in this research are described below:

3.4.1.1 Logistic Regression

Logistic Regression work on the pattern of supervised learning and calculate the probability of likely for the bug report input. Logistic regression finds the probability on Bernoulli distribution function and for the input to be transformed to the output for decision on sigmoid function.

$$p(y|x,w) = Bern(y|sigm(w^T x)) \qquad \text{[Eq. 3.1]}$$

$$sigm(\propto) \overset{\Delta}{=} \frac{1}{1+\exp(-\alpha)} \qquad \text{[Eq. 3.2]}$$

Here, Ber = Bernoulli function,

$Sigm$ = Sigmoid function,

y = label, x = Input features,

w = Model’s weight vector,

The model weight vector is calculated in the training phase of the model. In the testing or in the prediction phase the label with highest probability is assigned.

3.4.1.2 Support Vector Machine

Support Vector Machine (SVM) [32] is an another supervised, a popular and powerful machine learning method. SVM can be used both for classification and regression purpose but the primary purpose is to use for the classification purpose just like in this research. The objective of the SVM is to separate the n-dimensional into multiple classes with the best decision boundary. The decision boundary in the SVM is known as hyperplane and is shown in Fig. 6.

The hyperplane in the support vector machine is created with the help of the extreme points or the vectors and these points or the vectors are called support vectors and hence the name given to the support vector. The general diagram for the binary classification is sown in Fig. 6. Here in this research SVM is used for multi-class classification. Multi-class classification is the classification ins which we have more then two categories for the target.

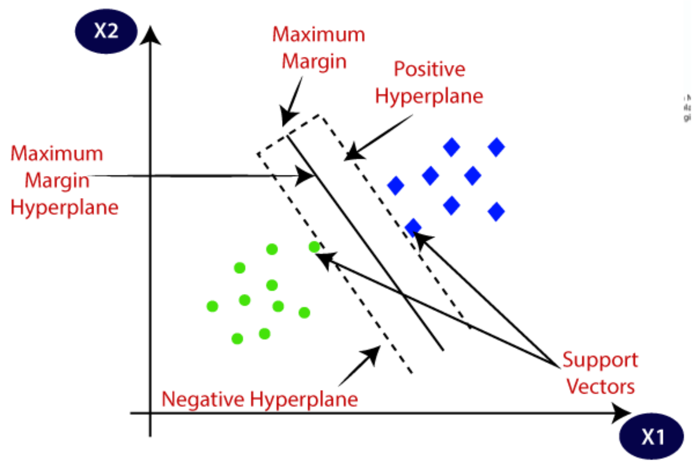


Figure 6 Support Vector Machine

3.4.1.3 Decision Tree

Just like SVM Decision Tree (DT) [32] also perform both regression and classification. Decision Tree is a powerful machine learning method for complex dataset and fit the complex dataset in a well handle manner. DT is the building block for the Random Forest which is available as the powerful machine learning algorithm. Decision Tree is built on the tree like pattern in which the leaves basically represent the classification classes while the branches represent the combination of the attributes which lead to the classification classes. Here in this research Decision Tree is used for the classification of both the Type and component classification. The generalized structure of the decision tree is shown in Fig. 8.

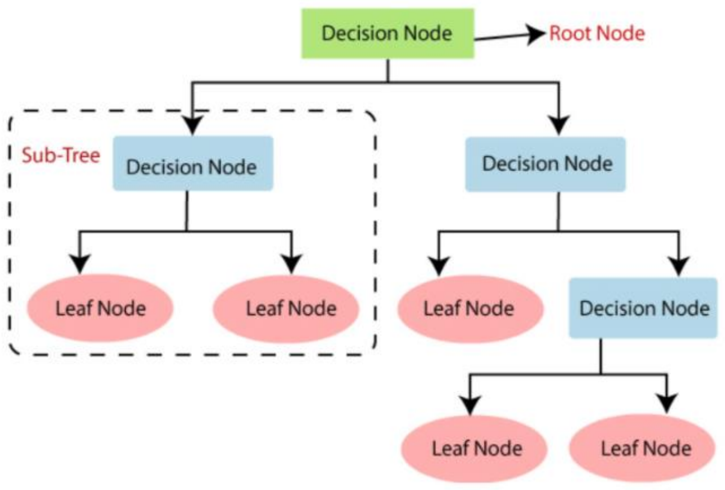


Figure 7 General Decision Tree Structure

3.4.1.4 Random Forest

Random Forest is based on the ensemble learning [32]. Ensemble learning is the combinations/aggregation of multiple predictors. Random Forest work in a manner of like training the model on multiple Decision Tree with different random sub set of the data set. The prediction is basically the aggregation of the predictions of each predictor. In a general perspective this model is a powerful model same as like if asking a question from 100 random people rather than one expert and aggregate the answer. In the same passion the class of the prediction is decided on the maximum number of votes came out from the multiple decision trees. The generalized form of Random Forest is shown in Fig. 8 below.

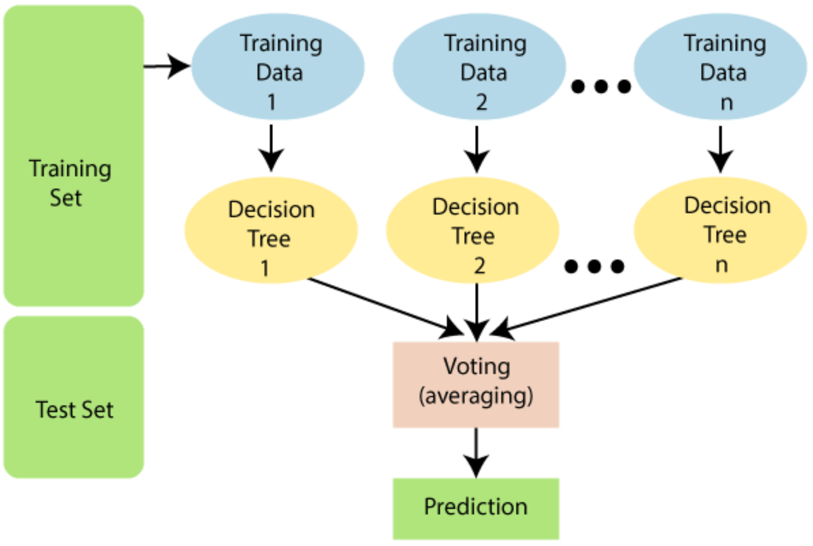


Figure 8 Random Forest General Form

3.4.1.5 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is the simplest classification method in machine learning where we have already defined classes resulted from the pattern in the dataset and new instances are classified based on these already defined classes with calculation of similarities. KNN is the supervised machine learning method and can be used for classification and regression. The prediction is decided on the Euclidean Distance and are decided by number of neighbors. KNN first find the distance for the new instance and then find the closest neighbors and finally vote for the labels as shown in Fig. 9.

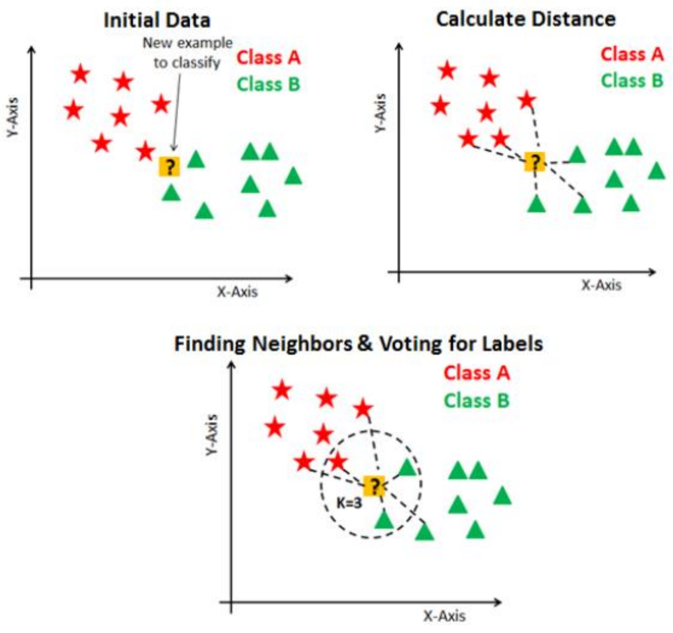


Figure 9 KNN General Steps

The above explained machine learning classifiers are implemented on the data from the pre-processing and performance is evaluated subsequently.

3.5 Performance Evaluation Metrics for Classification

For the purpose of model evaluation, certain metrics are used to compare the performance of the classifier on the testing data or the data in which there is no label provided. Metric is nothing but number we are interested in.

3.5.1 Accuracy

Accuracy [33] is the simplest and easy to calculate metric for the model evaluation. Accuracy calculates the correct predicted values. Accuracy is not a good metric for the classification models and in the case where there is unbalance of labels, the results become worst with the accuracy as the model will classify in the category where we have more instances. So, in the circumstances like that precision, recall and F-measure are preferred over accuracy.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \qquad \text{[Eq. 3.3]}$$

3.5.2 Precision

True positive (TP) is the category where the model predicts true in the true category while false positive (FP) is that the model predicts false for the true value. True negative (TN) is that the model predicts false for the false value and false negative (FN) is that the model predicts true for the false value. Precision [33] is hence called the positive predictive value and is given as:

$$precision = \frac{TP}{TP+FP} \qquad \text{[Eq. 3.4]}$$

3.5.3 Recall

Recall is known as sensitivity and is the percentage of correctly predicted over all positive and can be calculated as:

$$Recall = \frac{TP}{TP+FN} \qquad \text{[Eq. 3.5]}$$

.

3.5.4 F-measure

F-measure is the hormonic mean of both precision and recall, and is the most comprehensive measure for the performance.

$$F-measure = \frac{2\times precision \times recall}{precision+recall} \qquad \text{[Eq. 3.6]}$$

4 RESULTS AND ANALYSIS

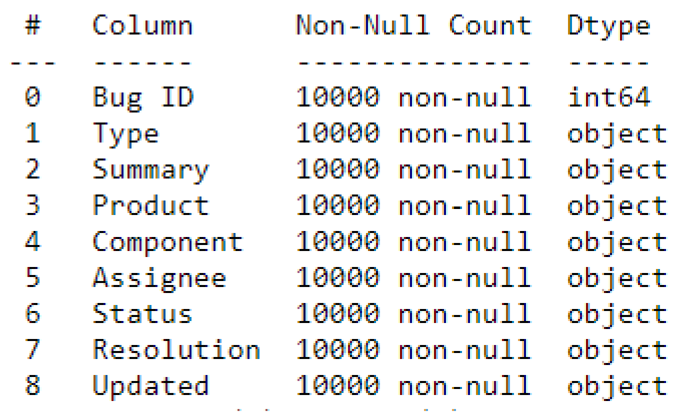
In this section result and analysis of the classification of the bug reports are explained in dept. As discussed in chapter 3 the whole method of this research, the classification is based on Firefox and ThunderBird dataset. We have a multi-class classification like we are classifying the bug on the type as well as on the component so that the developer team can quickly revolve the issue without consuming the time on the allotment of the bug to their respective department. First classification on the basis of Type of Bug with the Firefox dataset is explained and then with the Thunderbird data set. Classification on the basis of Component with Firefox and then with ThunderBird is explained in detail with the analysis.

All the results are generated in Python Jupyter Notebook with the input dataset as CSV file.

4.1 Bug Type Classification (Firefox Dataset)

We have three types that are defect, enhancement and task. We used many classifiers for the purpose of classification and all the results generated with the machine classifiers are discussed below. Below here the details of Firefox dataset with a bit information of datatype and null values.

Table 5 Firefox Dataset Columns



```
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Bug ID      10000 non-null  int64 
 1   Type        10000 non-null  object
 2   Summary     10000 non-null  object
 3   Product     10000 non-null  object
 4   Component   10000 non-null  object
 5   Assignee    10000 non-null  object
 6   Status      10000 non-null  object
 7   Resolution  10000 non-null  object
 8   Updated     10000 non-null  object
```

4.1.1 Support Vector Classifier

Fig. 10 shows the confusion matrix for the bug type classification which shows 1299 defect, 255 task and 20 enhancement classified accurately, while the rest are misclassified. In the figure below row indicate the True vs column indicate predicted labels and the combination of rows vs columns indicates TP, FP, FN and TN respectively.

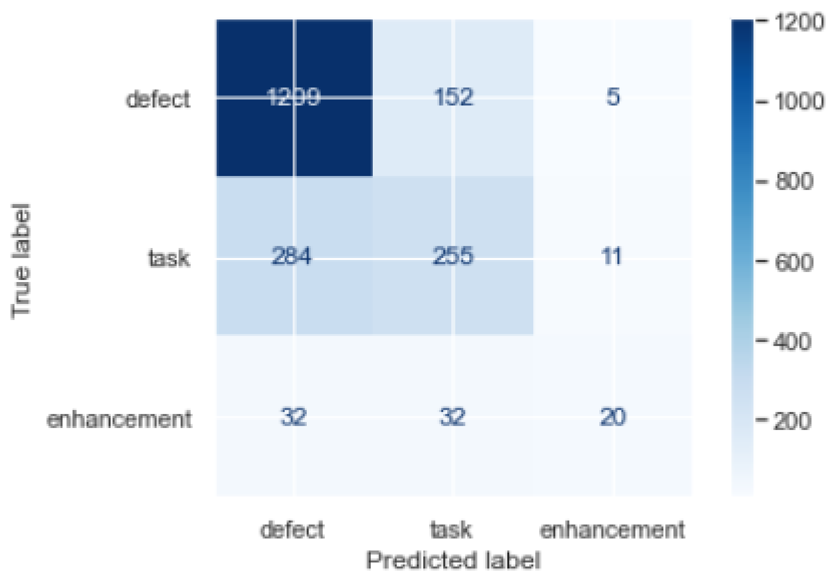


Figure 10 Confusion Matrix - SVC

The classfiication report of the Support Vector Classifer is shown in Fig. 11

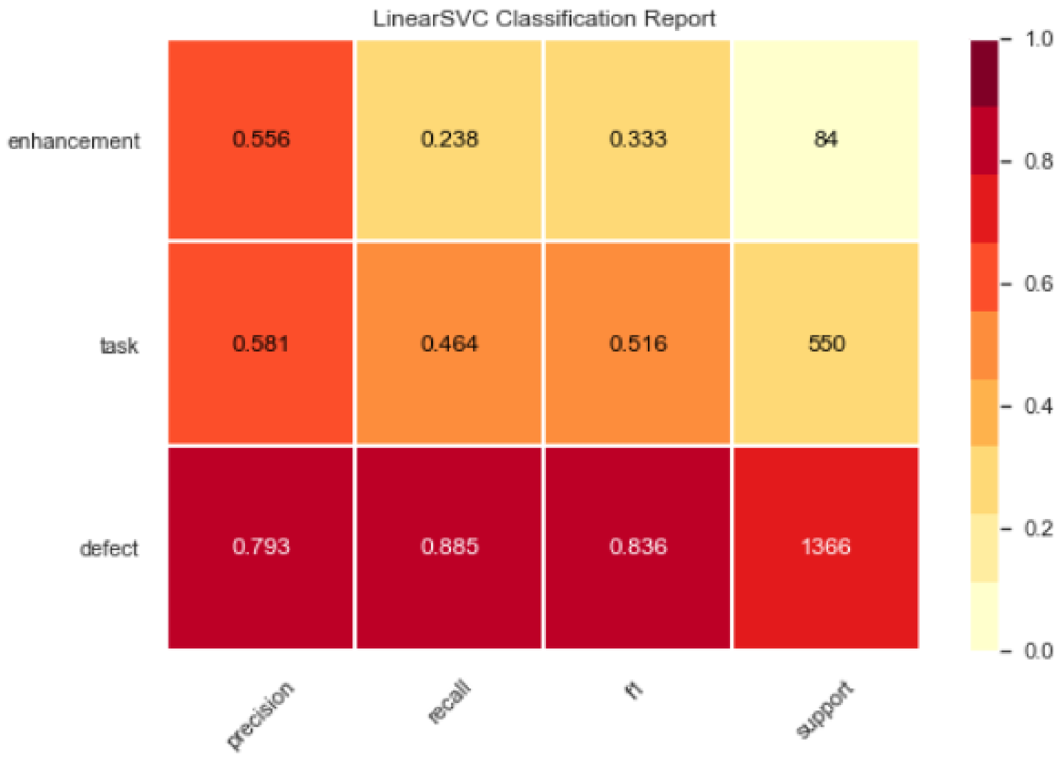


Figure 11 Linear SVC Classification Report

In Fig. 11 it shows the classifier, have better results for defect with having the precision of 0.793, recall of 0.836 and f-measure of 0.836. This model has overall accuracy of **74. 2%.**

4.1.2 Logistic Regression

Confusion Matrix (CM) for the logistic regression is shown in Fig. 11.

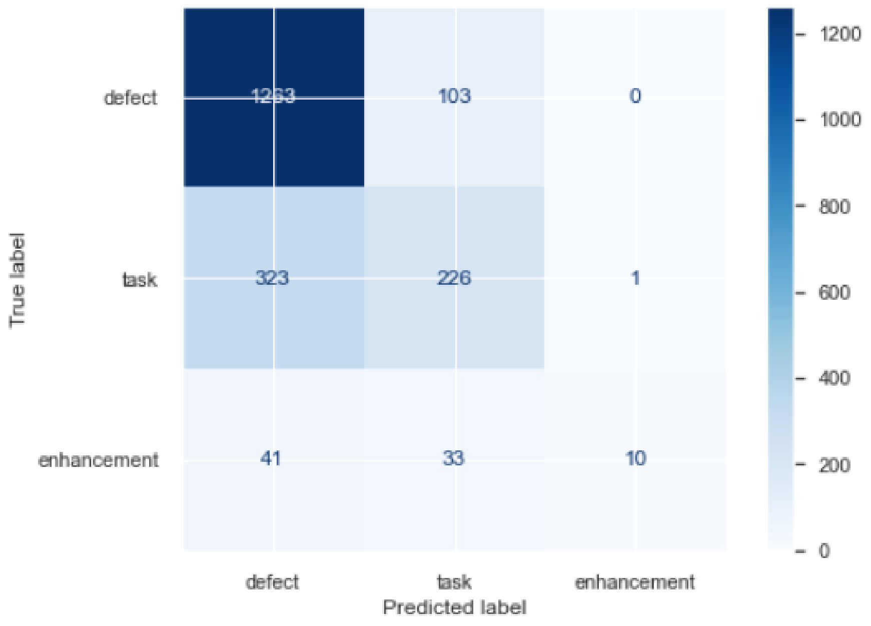


Figure 12 Confusion Matrix - LR

The classification report of Logistic Regression is shown in Fig. 12

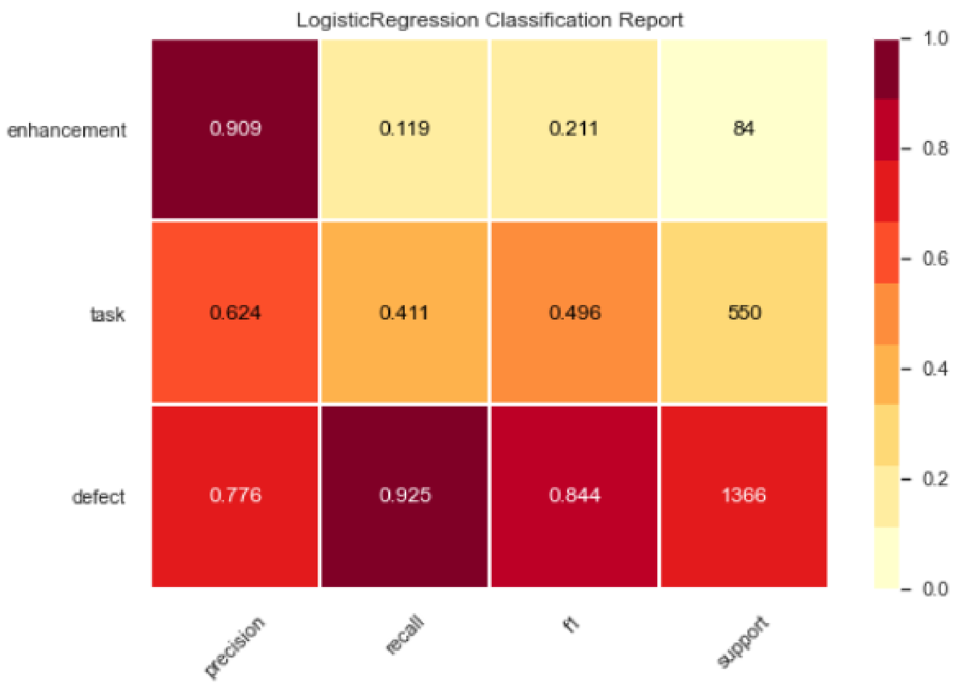


Figure 13 Classification Report - LR

In this model enhancement type has the precision of 0.909 while the defect has the recall of 0.925, while the F-measure for the defect is 0.844. This model has the overall accuracy of **75%.**

4.1.3 Decision Tree Classifier

The confusion Matrix for DT is shown in Fig. 14 below.

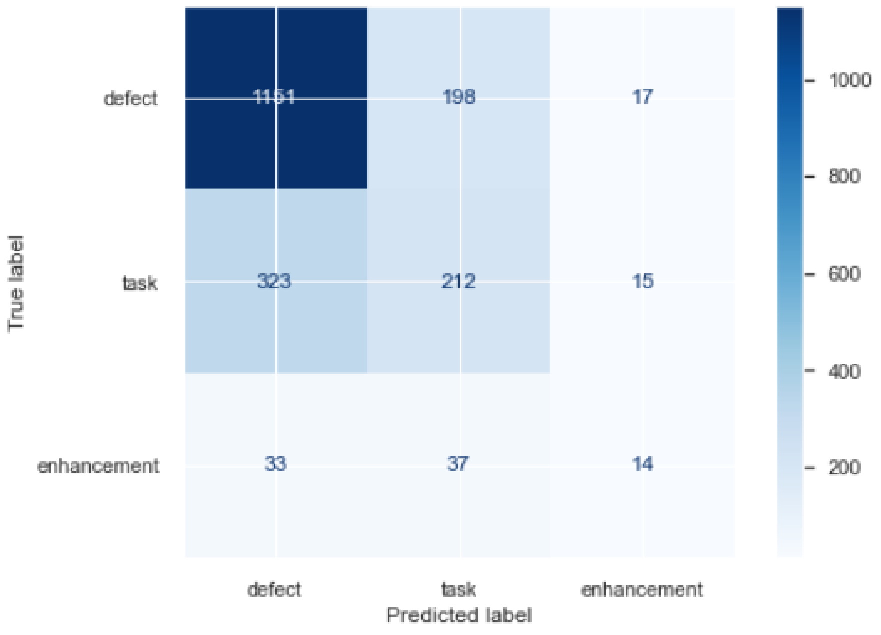


Figure 14 Confusion Matrix - DT

The classification report of DT model is shown in Fig. 15.

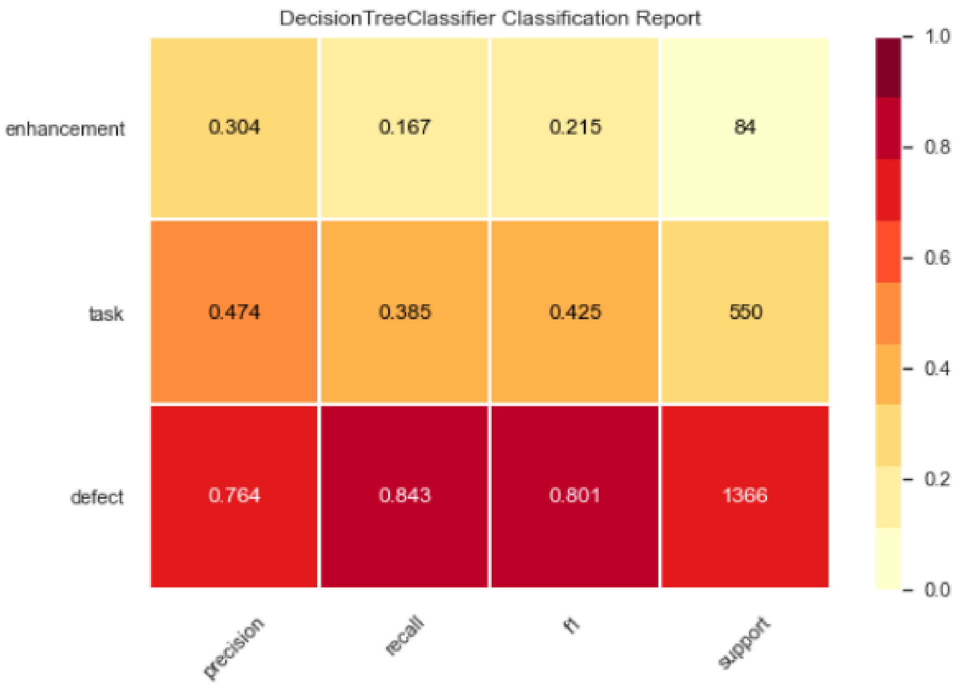


Figure 15 Classification Report - DT

This model shows the precision of 0.764 for defect and the recall of 0.843 for defect as well with the F-measure of 0.801. The accuracy of this model is **68.85**%.

4.1.4 Random Forest Classifier

Confusion Matrix is shown in Fig. 16.

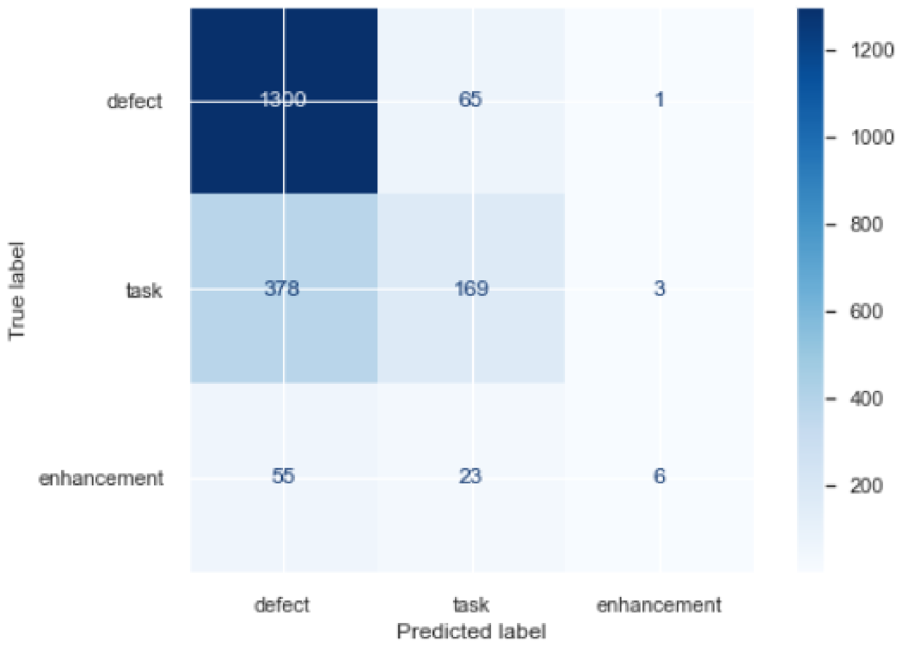


Figure 16 Confusion Matrix - RF

The classification report of this model is shown below.

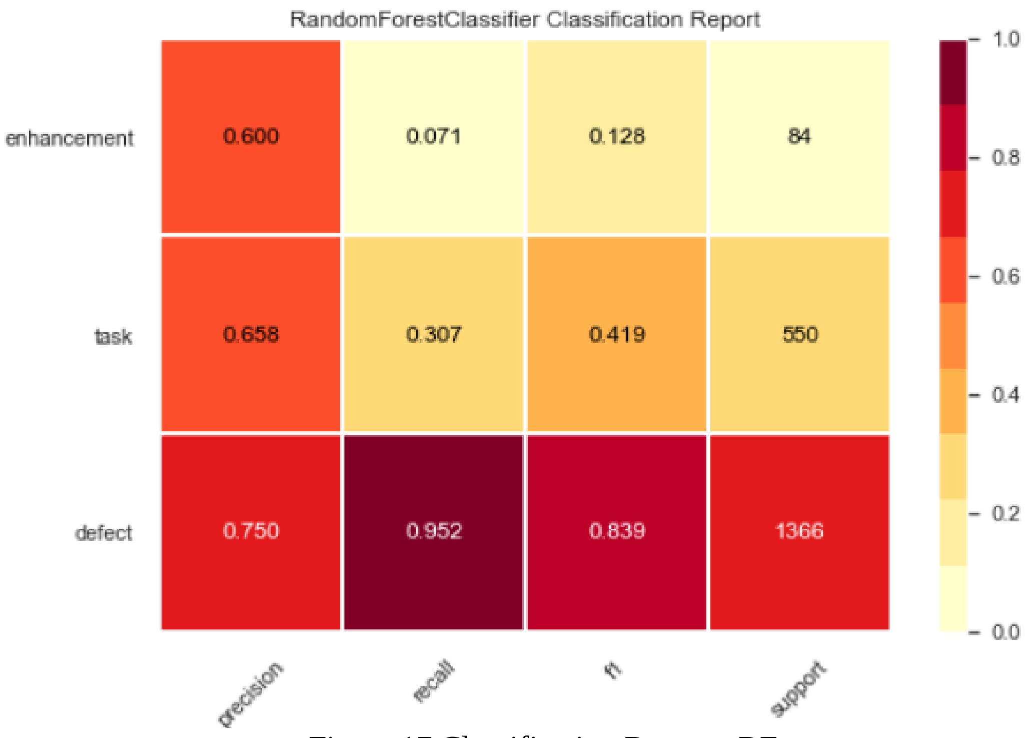


Figure 17 Classification Report - RF

Precision of this model is 0.75 for defect and recall of 0.952 with f1 score of 0.839. This model the overall accuracy of 74%.

4.1.5 K-Nearest Neighbors

Confusion Matrix of KNN is shown in Fig. 18.

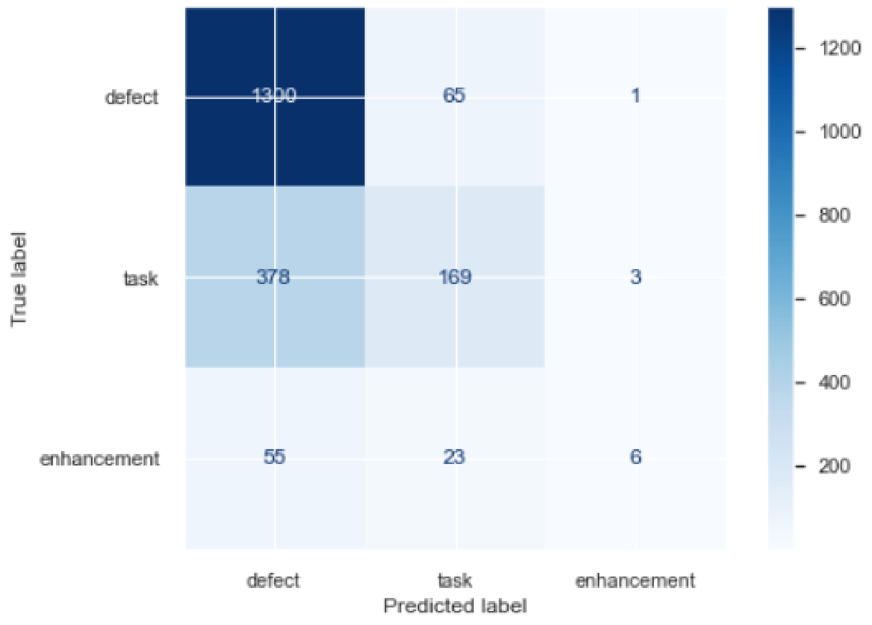


Figure 18 Confusion Matrix - KNN

Classification report of KNN classifier is shown in Fig. 19

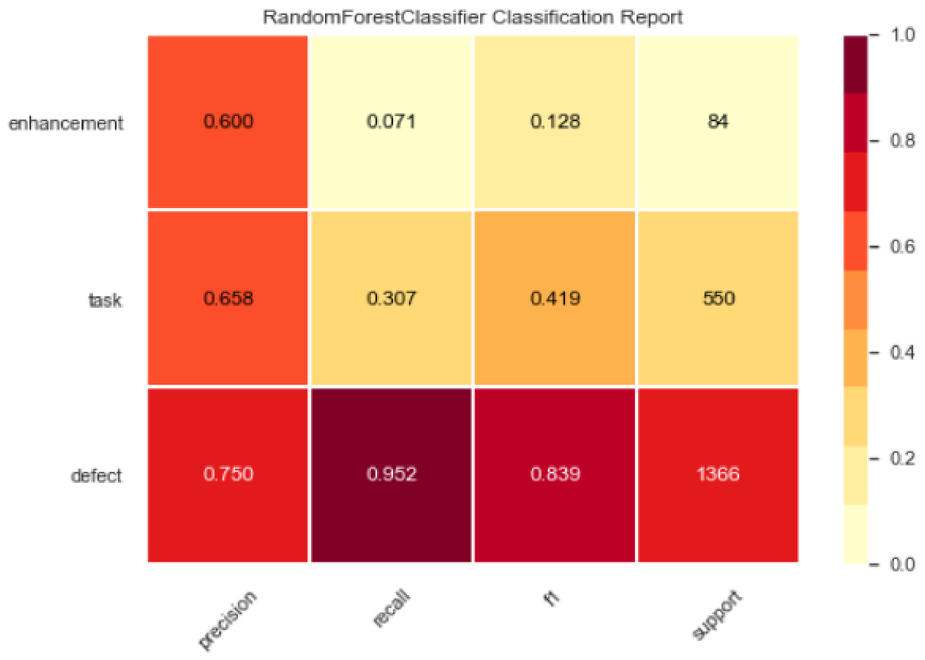


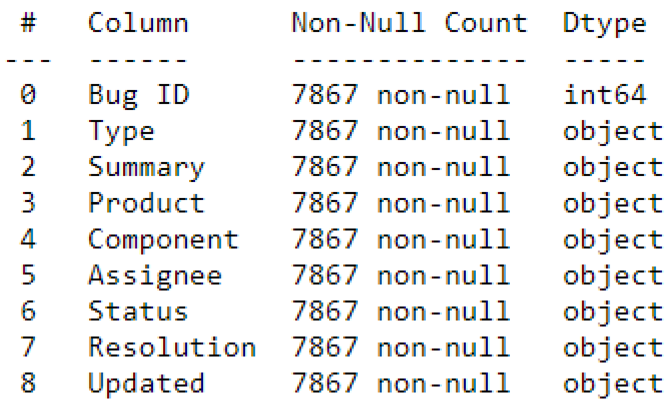
Figure 19 Classification Report - KNN

This model has the precision of 0.750 for defect with recall of 0.952 and f1 score of 0.839 and the overall accuracy for this model is **73.75%.**

4.2 Bug Type Classification (Thunderbird Dataset)

Thunderbird dataset have three types of bugs which are defect, enhancement and task. We implemented many classifiers for the purpose of classification and all the results generated with the machine classifiers are discussed below. Below here the details of Thunderbird dataset with a bit information of datatype and null values.

Table 6 Thunderbird Dataset Columns



```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Bug ID      7867 non-null   int64
 1   Type        7867 non-null   object
 2   Summary     7867 non-null   object
 3   Product     7867 non-null   object
 4   Component   7867 non-null   object
 5   Assignee    7867 non-null   object
 6   Status      7867 non-null   object
 7   Resolution  7867 non-null   object
 8   Updated     7867 non-null   object
```

4.2.1 Support Vector Classifier

Confusion Matrix details for SVC are shown below.

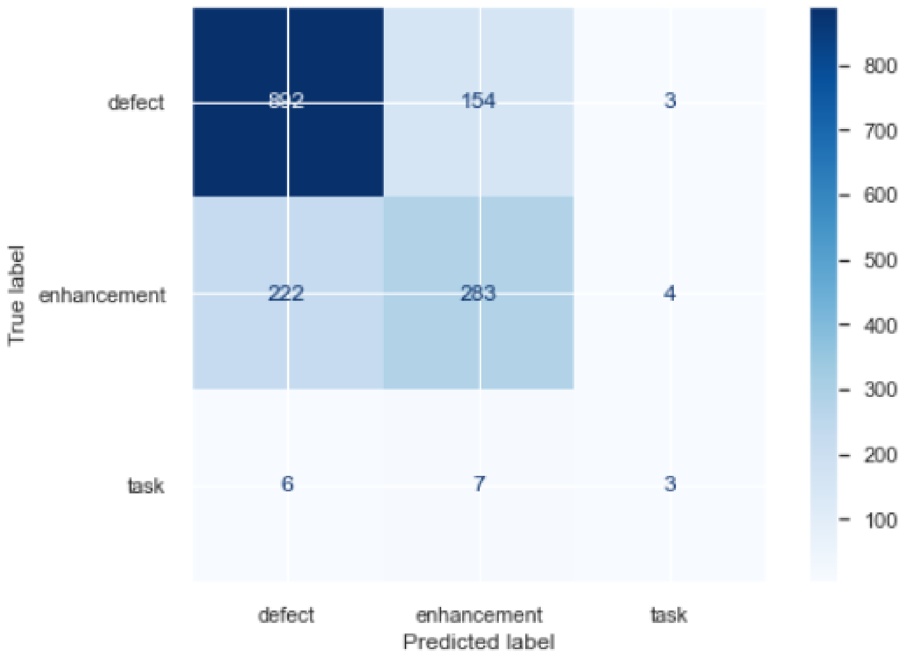


Figure 20 Confusion Matrix - SVC - TB

The classfiication report of the Support Vector Classifer is shown in Fig. 21

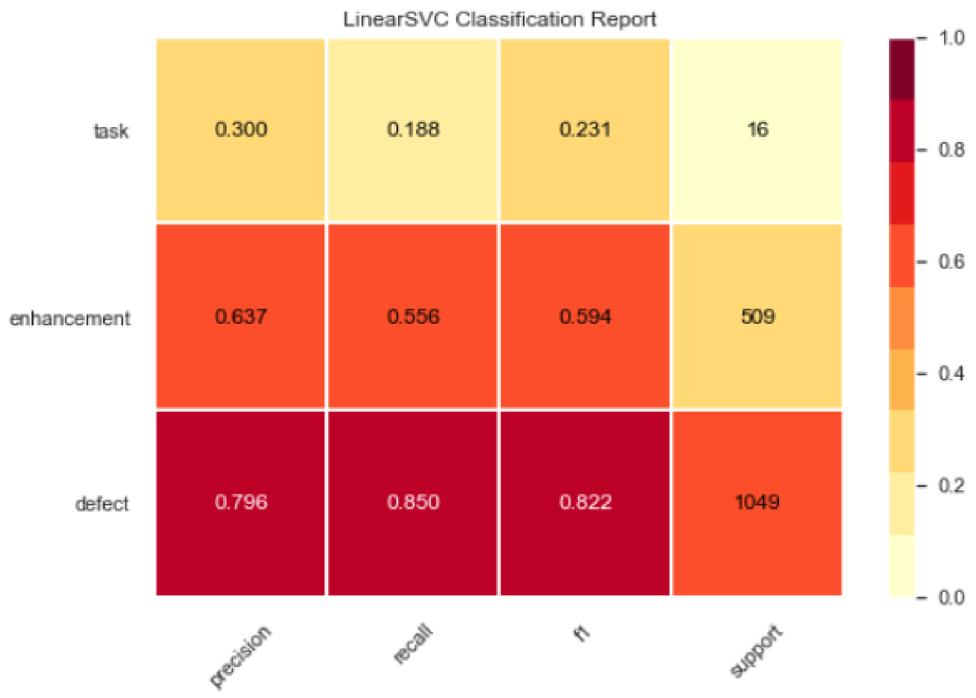


Figure 21 Linear SVC Classification Report – SVC - TB

In Fig. 21 it shows the classifier, have better results for defect with having the precision of 0.796, recall of 0.850 and f-measure of 0.822. This model has overall accuracy of **74.8%.**

4.2.2 Logistic Regression

Confusion Matrix (CM) for the logistic regression is shown in Fig. 22.

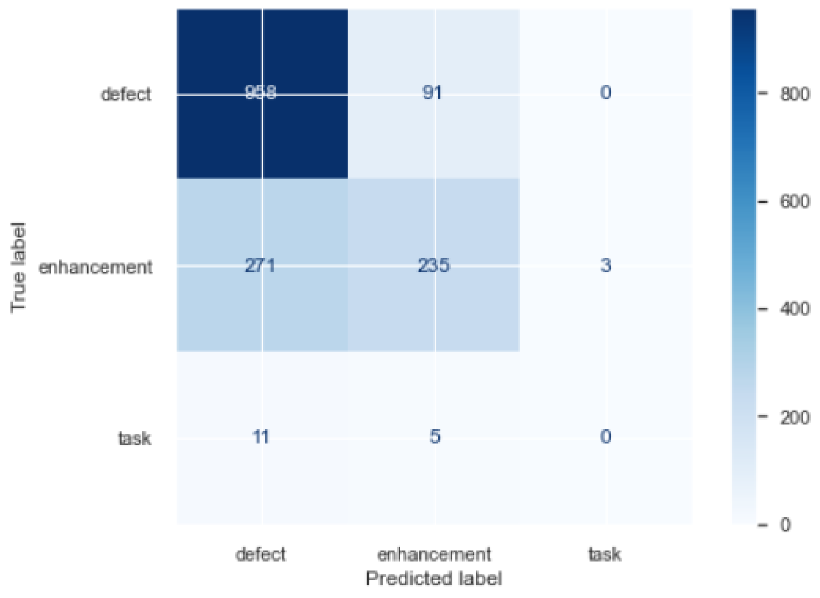


Figure 22 Confusion Matrix – LR - TB

The classification report of Logistic Regression is shown in Fig. 23

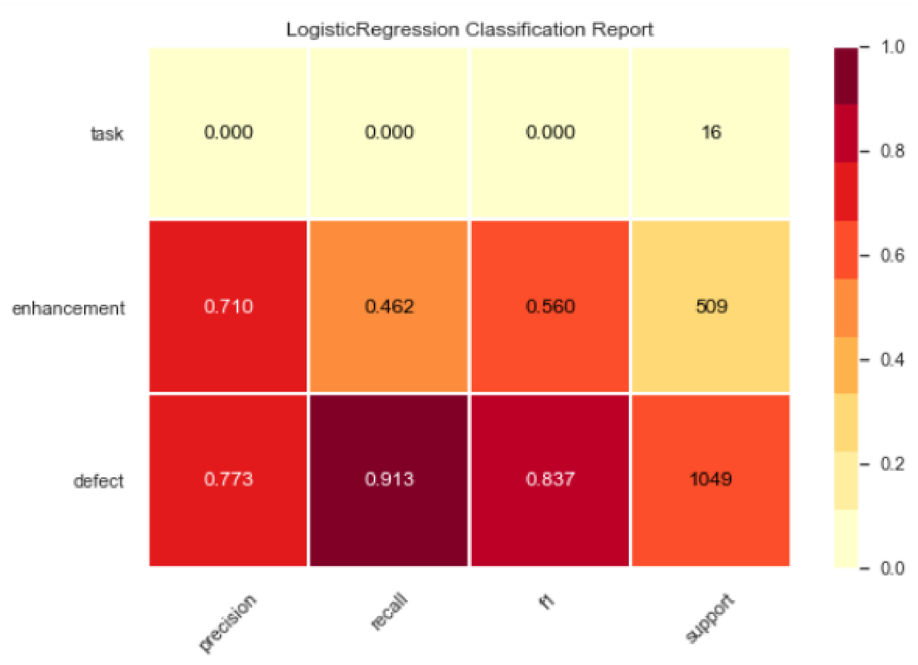


Figure 23 Classification Report – LR - TB

In this model defect type has the precision of 0.773 while the defect has the recall of 0.913, while the F-measure for the defect is 0.837. This model has the overall accuracy of **75.8%.**

4.2.3 Decision Tree Classifier

The confusion Matrix for DT is shown in Fig. 24 below.

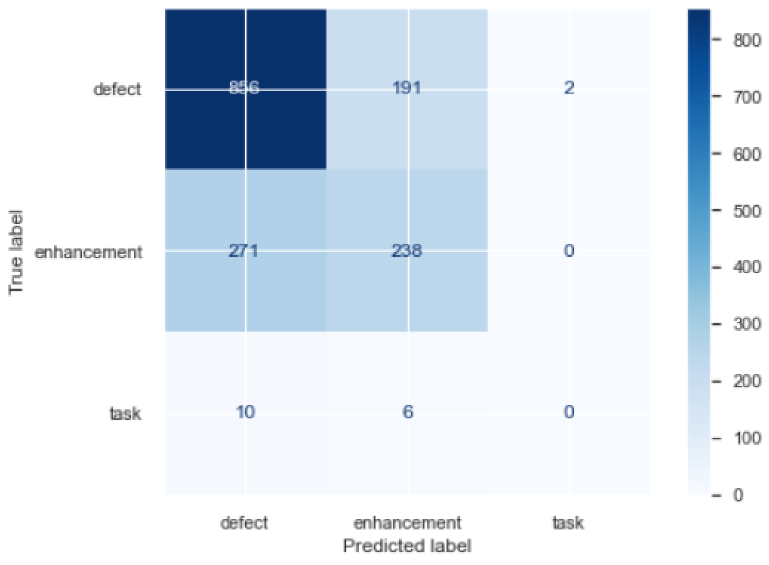


Figure 24 Confusion Matrix – DT - TB

The classification report of DT model is shown in Fig. 25.

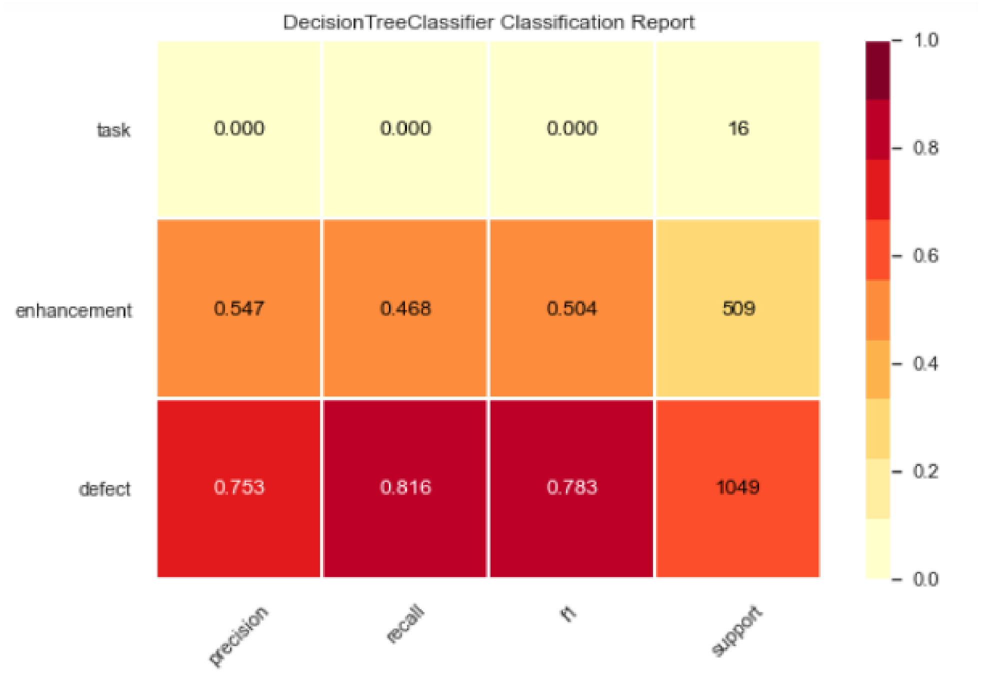


Figure 25 Classification Report – DT - TB

This model shows the precision of 0.753 for defect and the recall of 0.816 for defect as well with the F-measure of 0.783. The accuracy of this model is **69. 5**%.

4.2.4 Random Forest Classifier

Confusion Matrix is shown in Fig. 26.

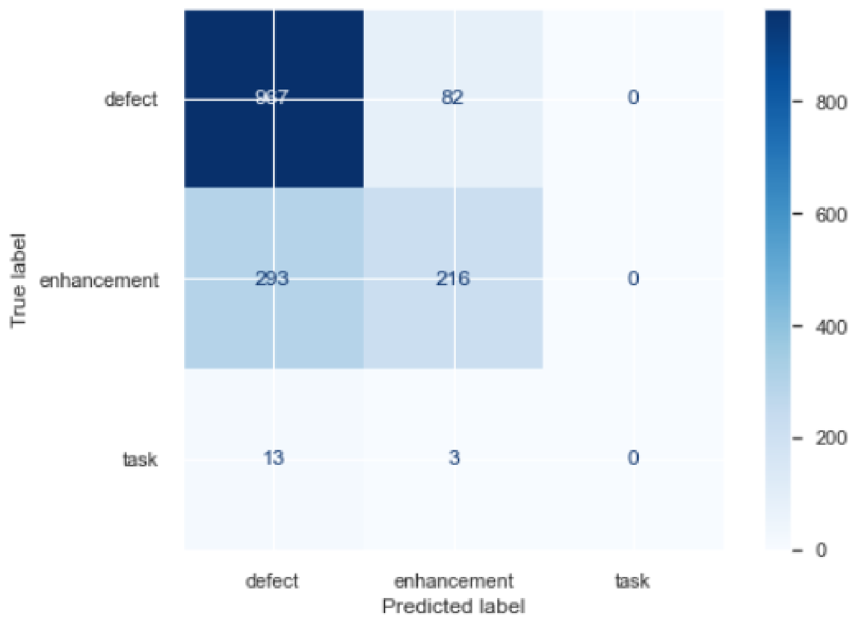


Figure 26 Confusion Matrix – RF - TB

The classification report of this model is shown below.

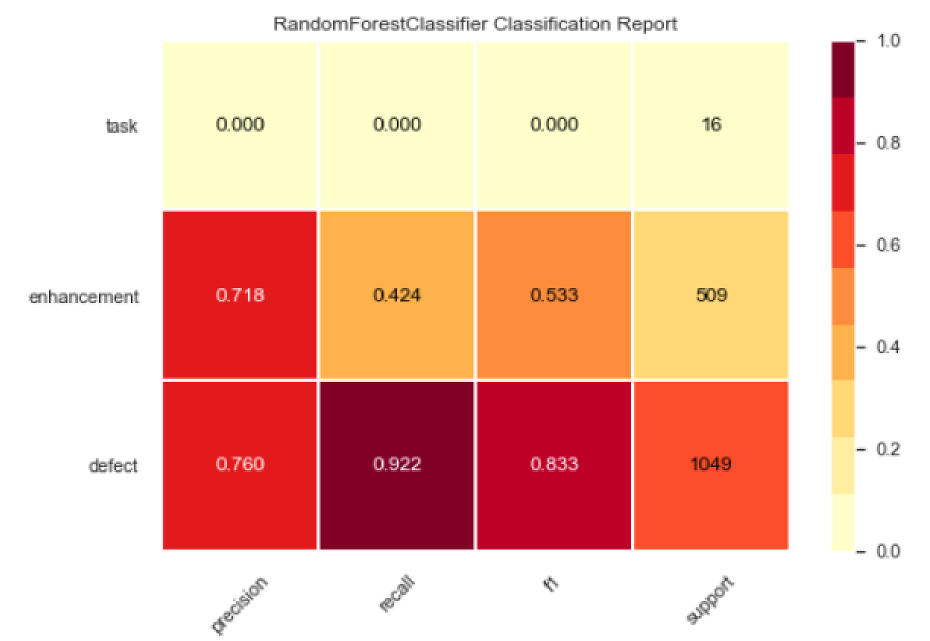


Figure 27 Classification Report – RF - TB

Precision of this model is 0.760 for defect and recall of 0.922 with f1 score of 0.833. This model the overall accuracy of 75.15%.

4.2.5 K-Nearest Neighbors

Confusion Matrix of KNN is shown in Fig. 28.

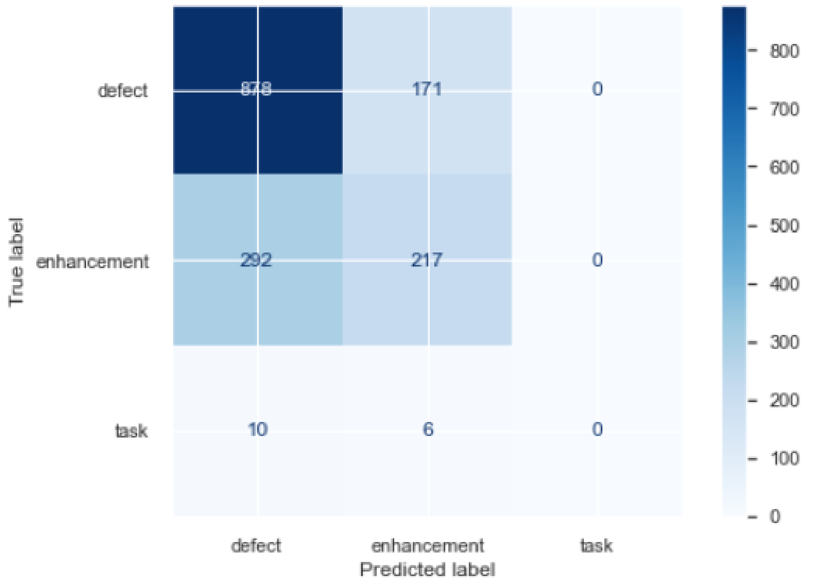


Figure 28 Confusion Matrix – KNN - TB

Classification report of KNN classifier is shown in Fig. 29

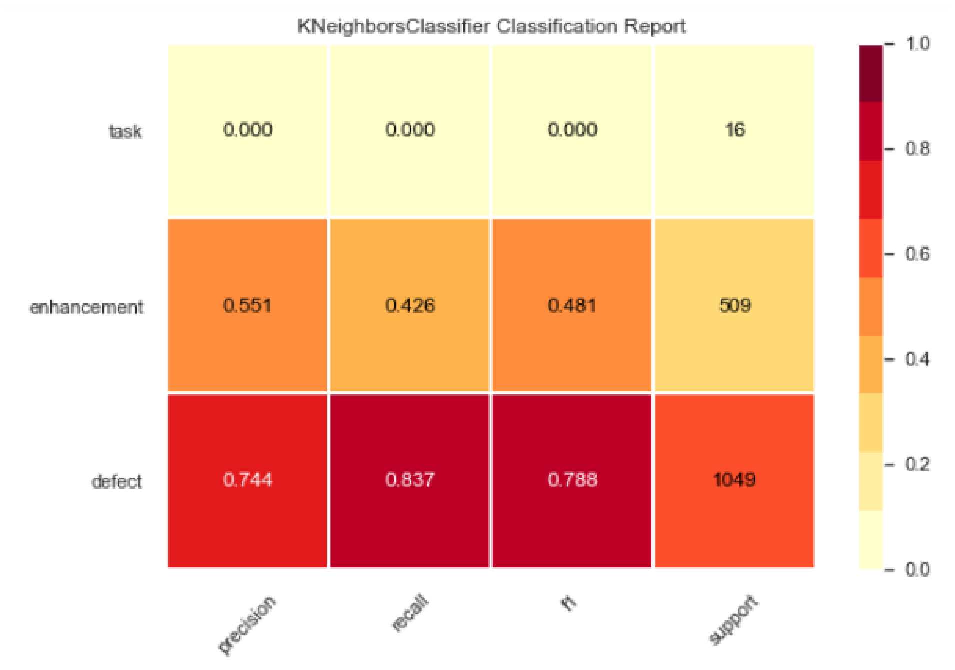


Figure 29 Classification Report - KNN

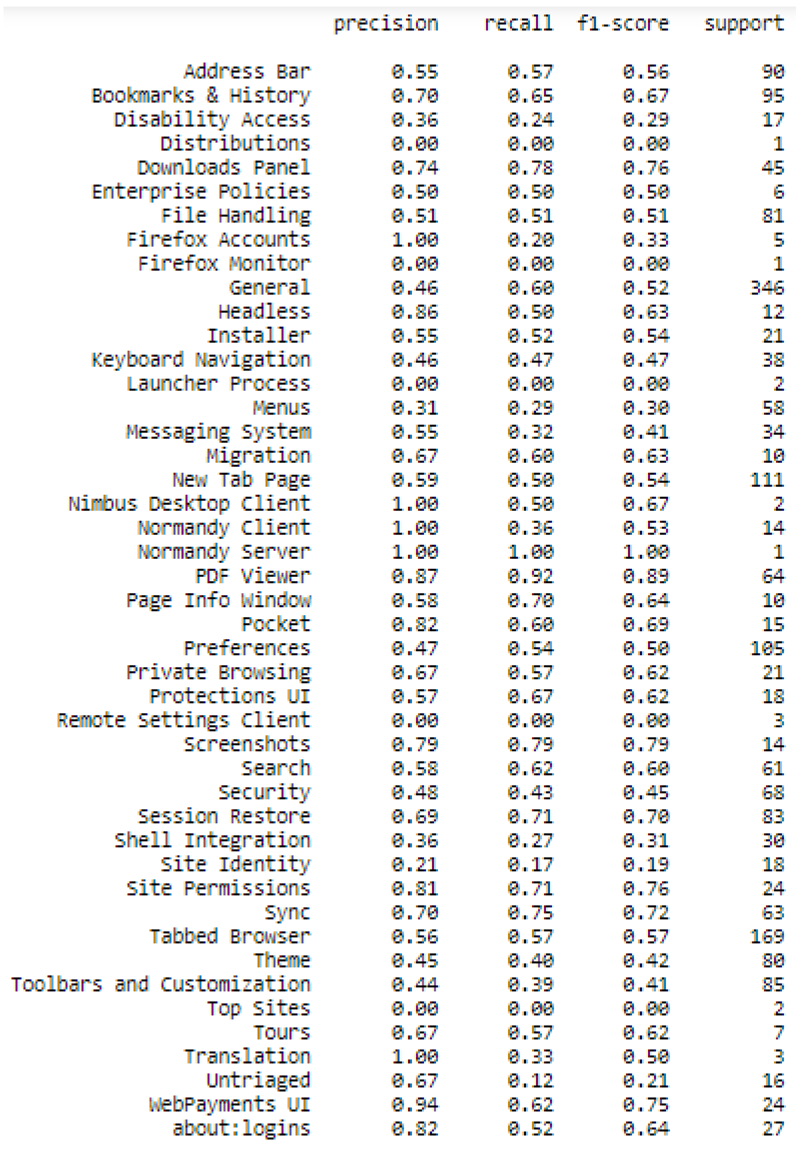
Precision of this model for defect is 0.744 with recall of 0.837 and f1 score of 0.788 and the overall accuracy for this model is **69.5%.**

4.3 Component Classification (Firefox Dataset)

As mentioned in the methodology chapter, component classification is also implemented in this research both on Firefox and Thunderbird data set. In Firefox data set we have 47 components as show in Table 2. All the classifiers are implemented for the component classification as well. Due to the large number of components the models performed in a well manner with achieving quite reasonable accuracy. On the type the models performed well. In the subsequent section all the model's performance and the classification reports are shown and analyzed as well. Here only SVC, Logistic regression and Random forest results are shown, while the rest are mentioned in the analysis. From the model performance perspectives Support Vector Classifier, Logistic regression and the Random Forest are promising in results on certain scenario in the current research.

4.3.1 Linear SVC

The classification report of this model is shown in Fig. 30 below.



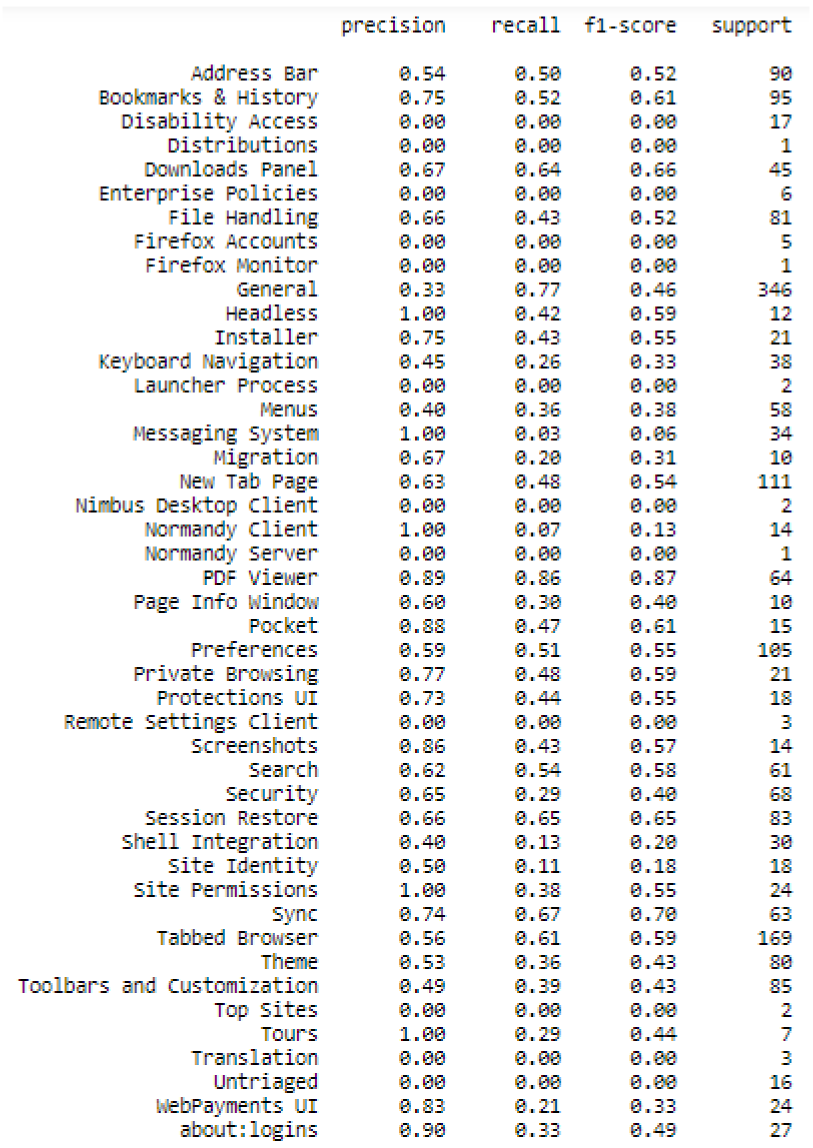
```
                          precision    recall  f1-score   support

              Address Bar      0.55      0.57      0.56        90
      Bookmarks & History      0.70      0.65      0.67        95
        Disability Access      0.36      0.24      0.29        17
            Distributions      0.00      0.00      0.00         1
          Downloads Panel      0.74      0.78      0.76        45
      Enterprise Policies      0.50      0.50      0.50         6
            File Handling      0.51      0.51      0.51        81
         Firefox Accounts      1.00      0.20      0.33         5
          Firefox Monitor      0.00      0.00      0.00         1
                  General      0.46      0.60      0.52       346
                 Headless      0.86      0.50      0.63        12
                Installer      0.55      0.52      0.54        21
      Keyboard Navigation      0.46      0.47      0.47        38
         Launcher Process      0.00      0.00      0.00         2
                    Menus      0.31      0.29      0.30        58
         Messaging System      0.55      0.32      0.41        34
                Migration      0.67      0.60      0.63        10
             New Tab Page      0.59      0.50      0.54       111
    Nimbus Desktop Client      1.00      0.50      0.67         2
          Normandy Client      1.00      0.36      0.53        14
          Normandy Server      1.00      1.00      1.00         1
               PDF Viewer      0.87      0.92      0.89        64
         Page Info Window      0.58      0.70      0.64        10
                   Pocket      0.82      0.60      0.69        15
              Preferences      0.47      0.54      0.50       105
         Private Browsing      0.67      0.57      0.62        21
           Protections UI      0.57      0.67      0.62        18
   Remote Settings Client      0.00      0.00      0.00         3
              Screenshots      0.79      0.79      0.79        14
                   Search      0.58      0.62      0.60        61
                 Security      0.48      0.43      0.45        68
          Session Restore      0.69      0.71      0.70        83
        Shell Integration      0.36      0.27      0.31        30
            Site Identity      0.21      0.17      0.19        18
         Site Permissions      0.81      0.71      0.76        24
                     Sync      0.70      0.75      0.72        63
           Tabbed Browser      0.56      0.57      0.57       169
                    Theme      0.45      0.40      0.42        80
Toolbars and Customization      0.44      0.39      0.41        85
                Top Sites      0.00      0.00      0.00         2
                    Tours      0.67      0.57      0.62         7
              Translation      1.00      0.33      0.50         3
                Untriaged      0.67      0.12      0.21        16
           WebPayments UI      0.94      0.62      0.75        24
             about:logins      0.82      0.52      0.64        27
```

Figure 30 Classification Report - SVC

This model performed well on certain components and showed quite reasonable precision, recall and f1- score like on the component bookmarks & history it acquired the precision of 0.70 with the recall value of 0.65 and f1-score of 0.67. This model has the overall accuracy of $\mathbf{55.15}\%$.

4.3.2 Logistic Regression

The classification report of this model is shown here:



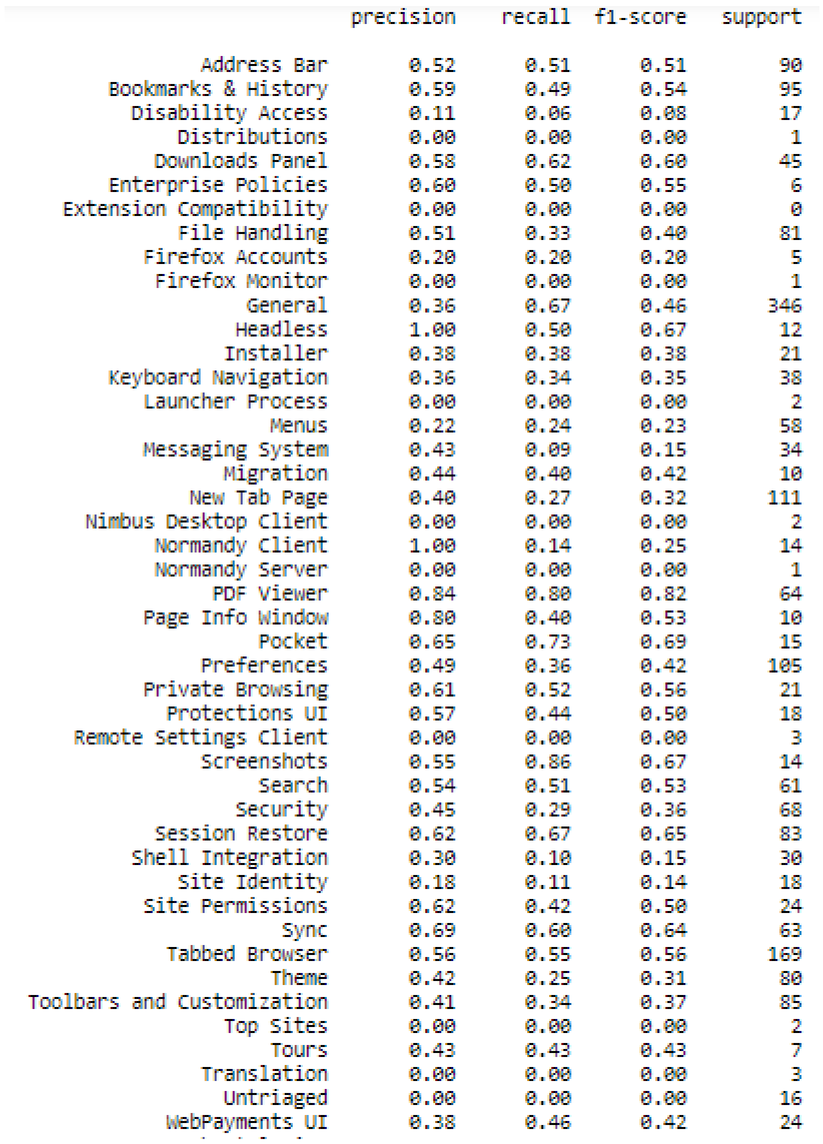
```
                          precision    recall  f1-score   support

              Address Bar       0.54      0.50      0.52        90
      Bookmarks & History       0.75      0.52      0.61        95
        Disability Access       0.00      0.00      0.00        17
            Distributions       0.00      0.00      0.00         1
          Downloads Panel       0.67      0.64      0.66        45
      Enterprise Policies       0.00      0.00      0.00         6
            File Handling       0.66      0.43      0.52        81
         Firefox Accounts       0.00      0.00      0.00         5
          Firefox Monitor       0.00      0.00      0.00         1
                  General       0.33      0.77      0.46       346
                 Headless       1.00      0.42      0.59        12
                Installer       0.75      0.43      0.55        21
      Keyboard Navigation       0.45      0.26      0.33        38
         Launcher Process       0.00      0.00      0.00         2
                    Menus       0.40      0.36      0.38        58
         Messaging System       1.00      0.03      0.06        34
                Migration       0.67      0.20      0.31        10
             New Tab Page       0.63      0.48      0.54       111
    Nimbus Desktop Client       0.00      0.00      0.00         2
          Normandy Client       1.00      0.07      0.13        14
          Normandy Server       0.00      0.00      0.00         1
               PDF Viewer       0.89      0.86      0.87        64
         Page Info Window       0.60      0.30      0.40        10
                   Pocket       0.88      0.47      0.61        15
              Preferences       0.59      0.51      0.55       105
         Private Browsing       0.77      0.48      0.59        21
           Protections UI       0.73      0.44      0.55        18
   Remote Settings Client       0.00      0.00      0.00         3
              Screenshots       0.86      0.43      0.57        14
                   Search       0.62      0.54      0.58        61
                 Security       0.65      0.29      0.40        68
          Session Restore       0.66      0.65      0.65        83
        Shell Integration       0.40      0.13      0.20        30
            Site Identity       0.50      0.11      0.18        18
         Site Permissions       1.00      0.38      0.55        24
                     Sync       0.74      0.67      0.70        63
           Tabbed Browser       0.56      0.61      0.59       169
                    Theme       0.53      0.36      0.43        80
Toolbars and Customization       0.49      0.39      0.43        85
                Top Sites       0.00      0.00      0.00         2
                    Tours       1.00      0.29      0.44         7
              Translation       0.00      0.00      0.00         3
                Untriaged       0.00      0.00      0.00        16
           WebPayments UI       0.83      0.21      0.33        24
             about:logins       0.90      0.33      0.49        27
```

Figure 31 Classification Report - LR

This model also performed well on certain component classification just like PDF viewer and having the precision of 0.89, with the recall 0.86 and the f1-score of 0.87 which is quire promising results. This model has the overall accuracy of **50.8**%.

4.3.3 Random Forest Classifier

The classification report of this model is given herein:



```
                           precision    recall  f1-score   support

               Address Bar       0.52      0.51      0.51        90
       Bookmarks & History       0.59      0.49      0.54        95
         Disability Access       0.11      0.06      0.08        17
             Distributions       0.00      0.00      0.00         1
           Downloads Panel       0.58      0.62      0.60        45
       Enterprise Policies       0.60      0.50      0.55         6
   Extension Compatibility       0.00      0.00      0.00         0
             File Handling       0.51      0.33      0.40        81
          Firefox Accounts       0.20      0.20      0.20         5
           Firefox Monitor       0.00      0.00      0.00         1
                   General       0.36      0.67      0.46       346
                  Headless       1.00      0.50      0.67        12
                 Installer       0.38      0.38      0.38        21
       Keyboard Navigation       0.36      0.34      0.35        38
          Launcher Process       0.00      0.00      0.00         2
                     Menus       0.22      0.24      0.23        58
          Messaging System       0.43      0.09      0.15        34
                 Migration       0.44      0.40      0.42        10
              New Tab Page       0.40      0.27      0.32       111
     Nimbus Desktop Client       0.00      0.00      0.00         2
           Normandy Client       1.00      0.14      0.25        14
           Normandy Server       0.00      0.00      0.00         1
                PDF Viewer       0.84      0.80      0.82        64
          Page Info Window       0.80      0.40      0.53        10
                    Pocket       0.65      0.73      0.69        15
               Preferences       0.49      0.36      0.42       105
          Private Browsing       0.61      0.52      0.56        21
            Protections UI       0.57      0.44      0.50        18
    Remote Settings Client       0.00      0.00      0.00         3
               Screenshots       0.55      0.86      0.67        14
                    Search       0.54      0.51      0.53        61
                  Security       0.45      0.29      0.36        68
           Session Restore       0.62      0.67      0.65        83
         Shell Integration       0.30      0.10      0.15        30
             Site Identity       0.18      0.11      0.14        18
          Site Permissions       0.62      0.42      0.50        24
                      Sync       0.69      0.60      0.64        63
            Tabbed Browser       0.56      0.55      0.56       169
                     Theme       0.42      0.25      0.31        80
Toolbars and Customization       0.41      0.34      0.37        85
                 Top Sites       0.00      0.00      0.00         2
                     Tours       0.43      0.43      0.43         7
               Translation       0.00      0.00      0.00         3
                Untriaged        0.00      0.00      0.00        16
            WebPayments UI       0.38      0.46      0.42        24
```

Figure 32 Classification Report - RF

This model also performed well on the PDF viewer component and showed the precision of 0.84, 0.80 for recall and f1-score of 0.82. This model has the overall accuracy of $\mathbf{46.2}$ %.

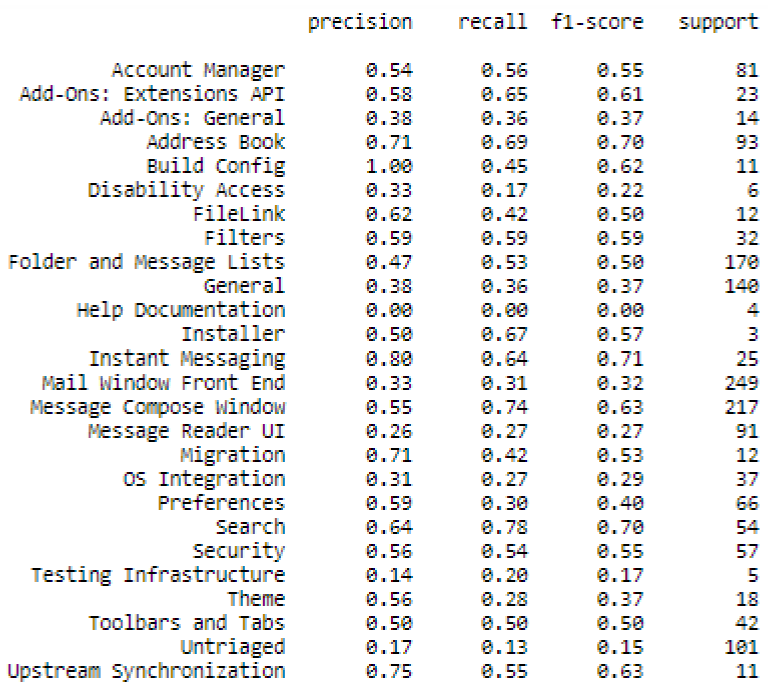
The above results were generated with Firefox dataset, below here are the results with the Thunderbird dataset.

4.4 Component Classification (Thunderbird Dataset)

Like Firefox component classification, Thunderbird dataset component are also classified in the same manner with using the same models. Same as the Firefox data set, this dataset also has many components as show in Table 4. The total number of components in Thunderbird data set is 35 which made this problem is the multi-class classification system. The results are generated with multiple models and evaluated for better model on the precision, recall and F1- score along with the accuracy. The results with thunderbird dataset are presented with analysis below.

4.4.1 Support Vector Classifier

The classification report is shown in Fig. 33.



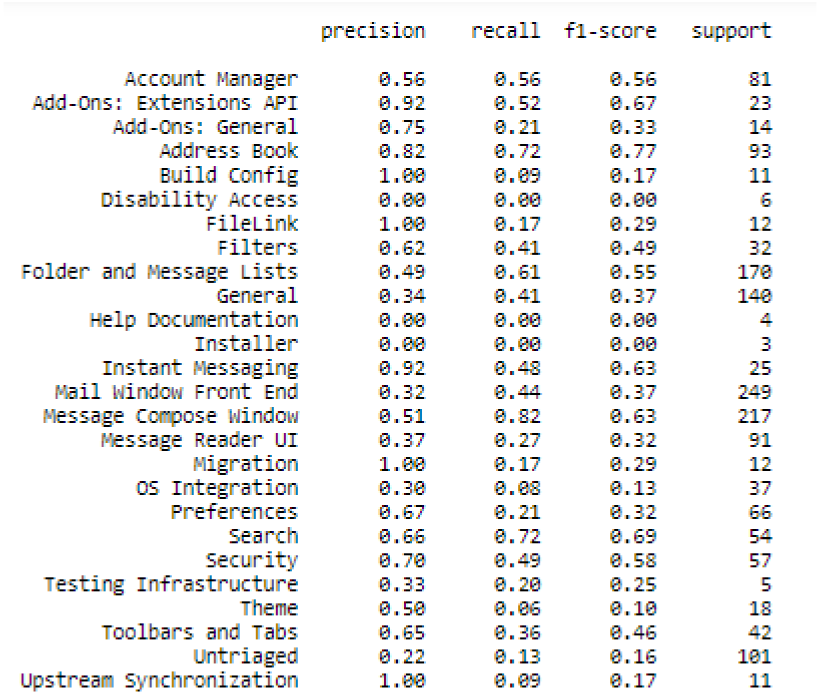
```
                          precision    recall  f1-score   support

         Account Manager       0.54      0.56      0.55        81
   Add-Ons: Extensions API     0.58      0.65      0.61        23
       Add-Ons: General        0.38      0.36      0.37        14
            Address Book       0.71      0.69      0.70        93
            Build Config       1.00      0.45      0.62        11
       Disability Access       0.33      0.17      0.22         6
                FileLink       0.62      0.42      0.50        12
                 Filters       0.59      0.59      0.59        32
  Folder and Message Lists     0.47      0.53      0.50       170
                 General       0.38      0.36      0.37       140
      Help Documentation       0.00      0.00      0.00         4
               Installer       0.50      0.67      0.57         3
       Instant Messaging       0.80      0.64      0.71        25
     Mail Window Front End     0.33      0.31      0.32       249
   Message Compose Window      0.55      0.74      0.63       217
       Message Reader UI       0.26      0.27      0.27        91
               Migration       0.71      0.42      0.53        12
          OS Integration       0.31      0.27      0.29        37
             Preferences       0.59      0.30      0.40        66
                  Search       0.64      0.78      0.70        54
                Security       0.56      0.54      0.55        57
   Testing Infrastructure      0.14      0.20      0.17         5
                   Theme       0.56      0.28      0.37        18
       Toolbars and Tabs       0.50      0.50      0.50        42
               Untriaged       0.17      0.13      0.15       101
  Upstream Synchronization     0.75      0.55      0.63        11
```

Figure 33 Classification Report - SVC - TB

The performance of this model is quite promising and showed some great results on certain components like on Build Config this model is showing the precision of 1.00 with the F1-score of 0.62. The overall efficiency of this model is $\mathbf{46.8}$%.

4.4.2 Logistic Regression

The classification reports is shown here for this model.



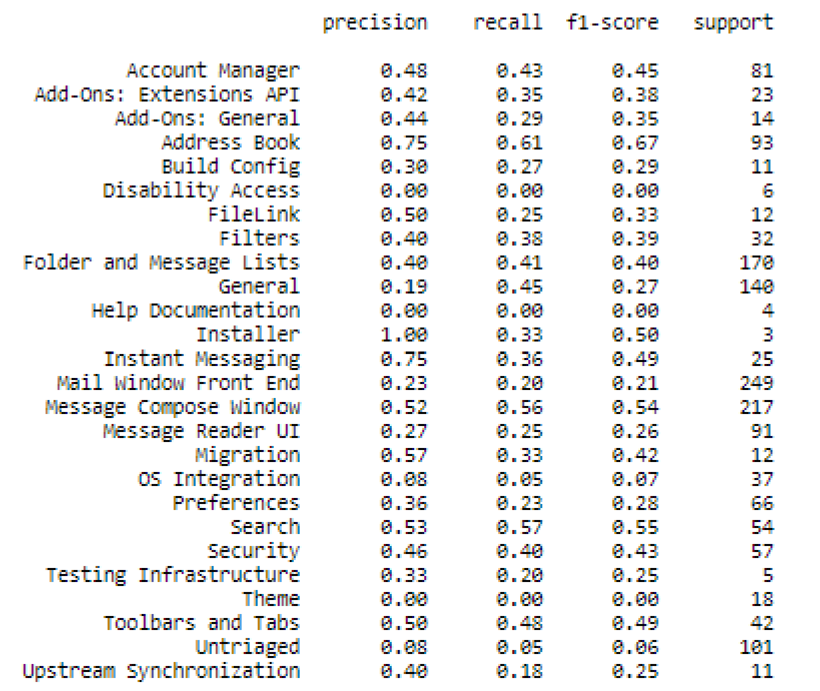
```
                          precision    recall  f1-score   support

         Account Manager       0.56      0.56      0.56        81
  Add-Ons: Extensions API       0.92      0.52      0.67        23
       Add-Ons: General       0.75      0.21      0.33        14
           Address Book       0.82      0.72      0.77        93
           Build Config       1.00      0.09      0.17        11
      Disability Access       0.00      0.00      0.00         6
               FileLink       1.00      0.17      0.29        12
                Filters       0.62      0.41      0.49        32
 Folder and Message Lists       0.49      0.61      0.55       170
                General       0.34      0.41      0.37       140
     Help Documentation       0.00      0.00      0.00         4
              Installer       0.00      0.00      0.00         3
      Instant Messaging       0.92      0.48      0.63        25
   Mail Window Front End       0.32      0.44      0.37       249
  Message Compose Window       0.51      0.82      0.63       217
      Message Reader UI       0.37      0.27      0.32        91
              Migration       1.00      0.17      0.29        12
         OS Integration       0.30      0.08      0.13        37
            Preferences       0.67      0.21      0.32        66
                 Search       0.66      0.72      0.69        54
               Security       0.70      0.49      0.58        57
  Testing Infrastructure       0.33      0.20      0.25         5
                  Theme       0.50      0.06      0.10        18
       Toolbars and Tabs       0.65      0.36      0.46        42
              Untriaged       0.22      0.13      0.16       101
 Upstream Synchronization       1.00      0.09      0.17        11
```

Figure 34 Classification Report - LR- TB

The overall accuracy of this model is **47.4**%.

4.4.3 Random Forest Classifier

Classification report for RF is show in Fig. 35.



```
                          precision    recall  f1-score   support

         Account Manager       0.48      0.43      0.45        81
  Add-Ons: Extensions API       0.42      0.35      0.38        23
       Add-Ons: General       0.44      0.29      0.35        14
           Address Book       0.75      0.61      0.67        93
           Build Config       0.30      0.27      0.29        11
      Disability Access       0.00      0.00      0.00         6
               FileLink       0.50      0.25      0.33        12
                Filters       0.40      0.38      0.39        32
 Folder and Message Lists       0.40      0.41      0.40       170
                General       0.19      0.45      0.27       140
     Help Documentation       0.00      0.00      0.00         4
              Installer       1.00      0.33      0.50         3
      Instant Messaging       0.75      0.36      0.49        25
   Mail Window Front End       0.23      0.20      0.21       249
  Message Compose Window       0.52      0.56      0.54       217
      Message Reader UI       0.27      0.25      0.26        91
              Migration       0.57      0.33      0.42        12
         OS Integration       0.08      0.05      0.07        37
            Preferences       0.36      0.23      0.28        66
                 Search       0.53      0.57      0.55        54
               Security       0.46      0.40      0.43        57
  Testing Infrastructure       0.33      0.20      0.25         5
                  Theme       0.00      0.00      0.00        18
       Toolbars and Tabs       0.50      0.48      0.49        42
              Untriaged       0.08      0.05      0.06       101
 Upstream Synchronization       0.40      0.18      0.25        11
```

Figure 35 Classification Report - RF - TB

The overall accuracy of this model is 47.27%.

5 DISCUSSION

1. **What information contained in user feedback are useful for bug reports?**

   For bug classification the words like ‘block access, ‘problem’, ‘issue’, ‘notable’, ‘doesn’t’, ‘not yet’, ‘unable’, ‘not found’, ‘lost’, ‘conflict’, ‘remove’, ‘resolve’, ‘severe’, ‘kill’, ‘unwanted’ etc.

2. which tool and techniques are reported to be most effective in overcoming the problems in bug report classification?

The tools used for the classification of bug reports which are showing promising results are Spacy library for the purpose of Natural Language Processing, with Sci-kit learn Library along with Jupyter Notebook. The techniques/algorithms that were implemented are showing promising results with better accuracy, precision and recall. Support Vector Machine, Logistic Regression and Random Forest are the best Classifier for the bug reports classification. With the comparison of these three Logistic Regression is effective in the classification of both the type and the components.

4. What algorithms will be most effective at filtering the user feedbacks and separating unrelated user feedbacks from the bug reports?

Natural Language Processing is showing the power of doing the filtration with much passion. SpaCy library is showing astonishing results rather the Natural Language Toolkit (NLTK).

5. How overcoming the bug report classification filtration problem would help in better release planning?

With best classification system, Developers will get their feedback from the users which will be processed in no time for finding the issues or the new features the users are interested, so this system will paly a key role in the release planning.

6 CONCLUSION AND FUTURE WORK

This chapter conclude the results and analysis of chapter 3 and also provide recommendation for the future work.

6.1 Conclusion

In this research with the help of Natural Language Processing, Machine Learning bugs reports are classified into their respective type and also classified on the basis of components. Due to this work the developers will get the classified bugs for the betterment of the industry by resolving the bug at the earliest and also will help the developers to add new features that are in demand of the users. Hence the time and cost of the developer industry will sharply reduce with this system and users retain ratio will abruptly increase as well. In this work both Firefox and Thunderbird dataset are used for the research purpose. Summary, Type and components features were used which were firstly preprocessed with spacy library by doing tokenization, stop words removal and lemmatization. With TF-IDF vectorization the features converted into vector with TF-IDF score for processing further to the machine learning models. Multiple machine learning algorithms were implemented like Support Vector Machine, Logistic Regression, Decision Tree, Random Forest and K-Nearest Neighbors. The performance of the models was evaluated with accuracy, precision, recall and F-measure. It was concluded that SVM, Logistic Regression and the Random Forest are showing promising results for the bug classification system. With reasonable comparison of the last three Logistic Regression is showing better results.

6.2 Future Work

The current work shows better results for the classification of the bug report but there is always a room for improvement in every model. These models can be further optimized by processing the textual data with alternate libraries and the machine learning algorithms can be replaced with the deep learning like recurrent neural network or the convolution neural network. These models can also be automated by integrating with the servers.

7 REFERENCES

[1] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, “The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-Driven Organization at Scale,” *Proc. - 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. ICSE 2017*, pp. 770–780, 2017, doi: 10.1109/ICSE.2017.76.

[2] W. Maalej, M. Nayebi, and G. Ruhe, “Data-Driven Requirements Engineering-An Update,” *Proc. - 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Pract. ICSE-SEIP 2019*, pp. 289–290, 2019, doi: 10.1109/ICSE-SEIP.2019.00041.

[3] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, “Toward data-driven requirements engineering,” *IEEE Softw.*, vol. 33, no. 1, pp. 48–54, 2016, doi: 10.1109/MS.2015.153.

[4] S. J. Dommati, R. Agrawal, R. M. R. G., and S. S. Kamath, “Bug Classification: Feature Extraction and Comparison of Event Model using Na\"ive Bayes Approach,” no. April 2013, 2013, [Online]. Available: http://arxiv.org/abs/1304.1677.

[5] C. Stanik and W. Maalej, “Requirements intelligence with OpenReq analytics,” *Proc. IEEE Int. Conf. Requir. Eng.*, vol. 2019-Septe, pp. 482–483, 2019, doi: 10.1109/RE.2019.00066.

[6] E. Giger, M. Pinzger, and H. Gall, “Predicting the fix time of bugs,” *Proc. - Int. Conf. Softw. Eng.*, pp. 52–56, 2010, doi: 10.1145/1808920.1808933.

[7] S. Kim, T. Zimmermann, E. J. Whitehead, and A. Zeller, “Predicting faults from cached history,” *Proc. - Int. Conf. Softw. Eng.*, pp. 489–498, 2007, doi: 10.1109/ICSE.2007.66.

[8] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, “Predicting the severity of a reported bug,” *Proc. - Int. Conf. Softw. Eng.*, pp. 1–10, 2010, doi: 10.1109/MSR.2010.5463284.

[9] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, “Predicting the location and number of faults in large software systems,” *IEEE Trans. Softw. Eng.*, vol. 31, no. 4, pp. 340–355, 2005, doi: 10.1109/TSE.2005.49.

[10] M. Lanza, M. Lanza, and R. Robbes, “An Extensive Comparison of <em>Bug</em> <em>Prediction</em> Approaches,” *Proc. Int. Conf. Min. Softw. Repos.*, pp. 31–41, 2003, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.175.5631.

[11] P. S. Kochhar, T. D. B. Le, and D. Lo, “It’s not a bug, it’s a feature: Does misclassification affect bug localization?,” *11th Work. Conf. Min. Softw. Repos. MSR 2014 - Proc.*, pp. 296–299, 2014, doi: 10.1145/2597073.2597105.

[12] S. Kim, H. Zhang, R. Wu, and L. Gong, “Dealing with noise in defect prediction,” *Proc. - Int. Conf. Softw. Eng.*, pp. 481–490, 2011, doi: 10.1145/1985793.1985859.

[13] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y. G. Guéhéneuc, “Is it a bug or an enhancement? A text-based approach to classify change requests,” *Proc. 2008 Conf. Cent. Adv. Stud. CASCON’08*, 2008, doi: 10.1145/1463788.1463819.

[14] T. Mens and S. Demeyer, “Software evolution,” *Softw. Evol.*, pp. 1–347, 2008, doi: 10.1007/978-3-540-76440-3.

[15] Y. Tian, D. Lo, and C. Sun, “DRONE: Predicting priority of reported bugs by multi-factor analysis,” *IEEE Int. Conf. Softw. Maintenance, ICSM*, pp. 200–209, 2013, doi: 10.1109/ICSM.2013.31.

[16] Y. Zhou, Y. Tong, R. Gu, and H. Gall, “Combining text mining and data mining for bug report classification,” *Proc. - 30th Int. Conf. Softw. Maint. Evol. ICSME 2014*, pp. 311–320, 2014, doi: 10.1109/ICSME.2014.53.

[17] S. Rastkar, G. C. Murphy, and G. Murray, “Automatic summarization of bug reports,” *IEEE Trans. Softw. Eng.*, vol. 40, no. 4, pp. 366–380, 2014, doi: 10.1109/TSE.2013.2297712.

[18] D. Behl, S. Handa, and A. Arora, “A bug Mining tool to identify and analyze security bugs using Naive Bayes and TF-IDF: A Comparative Analysis,” *ICROIT 2014 - Proc. 2014 Int. Conf. Reliab. Optim. Inf. Technol.*, pp. 294–299, 2014, doi: 10.1109/ICROIT.2014.6798341.

[19] P. Terdchanakul, H. Hata, P. Phannachitta, and K. Matsumoto, “Bug or not? Bug Report classification using N-gram IDF,” *Proc. - 2017 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2017*, no. September, pp. 534–538, 2017, doi: 10.1109/ICSME.2017.14.

[20] T. Zhang and B. Lee, “A bug rule based technique with feedback for classifying bug reports,” *Proc. - 11th IEEE Int. Conf. Comput. Inf. Technol. CIT 2011*, pp. 336–343, 2011, doi:

10.1109/CIT.2011.90.

[21] Neelofar, M. Y. Javed, and H. Mohsin, “An automated approach for software bug classification,” *Proc. - 2012 6th Int. Conf. Complex, Intelligent, Softw. Intensive Syst. CISIS 2012*, pp. 414–419, 2012, doi: 10.1109/CISIS.2012.132.

[22] C. Liu, J. Yang, L. Tan, and M. Hafiz, “R2Fix: Automatically generating bug fixes from bug reports,” *Proc. - IEEE 6th Int. Conf. Softw. Testing, Verif. Validation, ICST 2013*, pp. 282–291, 2013, doi: 10.1109/ICST.2013.24.

[23] A. Jayagopal, R. Kaushik, A. Krishnan, R. Nalla, and S. Ruttala, “Bug classification using machine and Deep Learning Algorithms,” *Mater. Today Proc.*, no. xxxx, pp. 1–5, 2021, doi: 10.1016/j.matpr.2021.01.188.

[24] N. Pingclasai, H. Hata, and K. I. Matsumoto, “Classifying bug reports to bugs and other requests using topic modeling,” *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2, pp. 13–18, 2013, doi: 10.1109/APSEC.2013.105.

[25] I. Chawla and S. K. Singh, “An automated approach for bug categorization using fuzzy logic,” *ACM Int. Conf. Proceeding Ser.*, vol. 18-20-Febr, pp. 90–99, 2015, doi: 10.1145/2723742.2723751.

[26] G. Indah, P. Sari, and D. O. Siahaan, “An Attribute Selection For Severity Level Determination According To The Support Vector Machine Classification Result,” pp. 305–310, 2011.

[27] N. K. S. Roy and B. Rossi, “Towards an improvement of bug severity classification,” *Proc. - 40th Euromicro Conf. Ser. Softw. Eng. Adv. Appl. SEAA 2014*, pp. 269–276, 2014, doi: 10.1109/SEAA.2014.51.

[28] G. Sharma, S. Sharma, and S. Gujral, “A Novel Way of Assessing Software Bug Severity Using Dictionary of Critical Terms,” *Procedia Comput. Sci.*, vol. 70, pp. 632–639, 2015, doi: 10.1016/j.procs.2015.10.059.

[29] N. Pandey, D. K. Sanyal, A. Hudait, and A. Sen, “Automated classification of software issue reports using machine learning techniques: an empirical study,” *Innov. Syst. Softw. Eng.*, vol. 13, no. 4, pp. 279–297, 2017, doi: 10.1007/s11334-017-0294-1.

[30] G. G. Chowdhury, “Natural language processing,” *Annu. Rev. Inf. Sci. Technol.*, vol. 37, pp. 51–89, 2003, doi: 10.1002/aris.1440370103.

[31] F. Sebastiani, “Machine Learning in Automated Text Categorization,” *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002, doi: 10.1145/505282.505283.

[32] B. Derviş, *Hands-on Machine Learning with Sklearn, keras & tensorflow*, vol. 53, no. 9. 2013.

[33] H. Van Halteren, W. Daelemans, and J. Zavrel, “Improving accuracy in word class tagging through the combination of machine learning systems,” *Comput. Linguist.*, vol. 27, no. 2, pp. 199–229, 2001, doi: 10.1162/089120101750300508.