T.C.

ALTINBAS UNIVERSITY

Institute of Graduate Studies

Information Technologies

HYBRID OPTIMIZATION OF MULTIPLE INTELLIGENT RECOMMENDATION ENGINES FOR SOFTWARE DEVELOPMENT CYCLES

Ali Jaafar Meera ALARKAWAZI

Master of Science

Supervisor

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Istanbul, 2021

HYBRID OPTIMIZATION OF MULTIPLE INTELLIGENT RECOMMENDATION ENGINES FOR SOFTWARE DEVELOPMENT CYCLES

by

Ali Jaafar Meera ALARKAWAZI

Information Technologies

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2021

The thesis titled “HYBRID OPTIMIZATION OF MULTIPLE INTELLIGENT RECOMMENDATION ENGINES FOR SOFTWARE DEVELOPMENT CYCLES” prepared and presented by “Ali Jaafar Meera ALARKAWAZI” was accepted as a Master of Science Thesis in Information Technologies.

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	School of Engineering and Architecture, Altinbas University	
Asst. Prof. Dr. Sefer KURNAZ	School of Engineering and Architecture, Altinbas University	
Prof. Dr. Adem KARAHOCA	Computer Engineering, MEF University	

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ali Jaafar Meera ALARKAWAZI

DEDICATION

This study is wholeheartedly dedicated to our beloved parents, who have been our source of inspiration and gave us strength when we thought of giving up, who continually provide their moral, spiritual, emotional, and financial support.

To our brothers, sisters, relatives, friends, and classmates who shared their words of advice and encouragement to finish this study.

ACKNOWLEDGEMENTS

We are really grateful because we managed to complete out thesis within the give time. This thesis could not be completed without the effort cooperation of our friends. We also thank my Asst. prof. Dr. Abdullahi Abdu IBRAHIM for guidance and encouragement in finishing this thesis and also for teaching me this thesis.

ABSTRACT

HYBRID OPTIMIZATION OF MULTIPLE INTELLIGENT RECOMMENDATION ENGINES FOR SOFTWARE DEVELOPMENT CYCLES

Ali Jaafar Meera ALARKAWZI

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Date: 09/2021

Pages: 63

The primary purpose is to create a hybrid recommendation system approach to improve the performance of such systems. This recommendation system would typically be used to assign or suggest a small number of developers suitable for troubleshooting a bug report. Managing collections inside bug repositories is software developers' task to fix any particular bugs that have been identified. Bugs are often created, so the number of developers needed is high, so it's hard to decide how many to assign to specific tasks.

This analysis aims better to understand the outcomes of the latest scientific methods. We also addressed developer prioritization and how it can be used to determine the assignment of a problem to a developer. We have studied two aspects: first, the selection of bug reports using hybrid machine learning methods, and modelling developer prioritization in the bug repository and supporting developer assignment tasks with our model. Second, we modelled the relevant objectives suggested by the developers' backgrounds based on proven knowledge and experience. The study focuses on two topers' experience with fixing bugs and developer rankings in the App Store. We've tried to take better assignments using developer prioritization

in bug repositories, e.g., bug triage, severity identification and re-opened bug prediction. We examine the output of the model in a representative sample of bug repositories. The results show that the prioritization of developers' prioritization triage worker and allow the program to solve the bugs more effectively in support of the software support has been clarified. After the introduction, Chapter 2 relates the literature; chapter 3 is about the methodologies used in studies, the findings and explanation are described in this thesis, chapter 4, the following chapter is the description accompanied by references.

Keywords: Open-source Bugs, Hybrid Intelligent optimization, random trees, Naive Bayes, artificial neural networks, genetic algorithm

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

AI	:	Artificial Intelligence
ANN	:	Artificial Neural Networks
API	:	Application Program Interface
CBF	:	Content Based Filtering
CF	:	Collaborative Filtering
GE	:	Genetic Algorithm
IDE	:	Integrated Development Environment
MIMD	:	Multiple Instruction Multiple Data
NB	:	Naïve Bayes
OOB	:	Out Of Bag
OS	:	Operating System
RITS	:	Recommendation Systems in the small
RS	:	Recommendation System
RSSE	:	Recommendation System for Software Engineering
SDLC	:	Software Development Life Cycle
SIMD	:	Single Instruction Multiple Data
SSR	:	Small Scale Recommendation
VCS	:	Version Control System

LIST OF SYMBOLS

C : Class label

dim : Dimension

E : Entropy

I : Impurity

K : Set of inputs

o : Sigmoid output

Pdf : Probability Density Function

P_r : Probability

T : Training set

w : Weight

x : Input

y : Output

1. INTRODUCTION

In software development cycles, open-source bug reports are used in the software development system as many as 60 percent of all bugs. The '10x Rule' – outlier bugs are often much harder to locate, and repair, than other bugs. Still, some bugs can't be detected or patched at all – or at least not with the knowledge we have about recognizing and fixing bugs. Moreover, the number of software bugs has significantly increased. Additionally, more than 180 bugs were identified in Eclipse's bug tracking system in the development period. Moreover, Debian created nearly 140 bug outstanding reports [1]. Bug reporting tools help developers identify bugs and work to fix them. In addition, for bug reports to be assigned as properly as possible is a challenge for the software development industry.

Open-source project developers also use an open bug repository for all bugs. The error report must be delegated to team members who are taking responsibility to fix the bug. When a new report comes in, there's a small group of developers who are expected to fix the bug. Therefore, this will help bug triage staff make decisions about how to prioritize fixing the bug study. Bug repositories are a form of issue tracking system that include a database of any hardware, software and programming issues [2]. The open-source bug repositories, as opposed to the closed-source commercial ones, as they are open to all for free access. These repositories play an important role in the collaboration between programmers and the project that enables it to work.

The benefits of using machine learning is that, as the algorithms will evolve with the project, the program will become more accurate. This also eliminates one of the difficulties of using human intelligence, as humans learn and attempt to develop a "good enough" heuristic package. The guidelines would apply to a wider range of projects as the heuristics for a given (possible partial) training group have not been established and the heuristics could adjust for new plans.

Many data mining systems are used so that different recommendations can be made. Technologies required to maintain and manage the RSSE (Software Engineering Recommendation Systems) can be costly in the device, computer network, and computer software. Recommendation frameworks that evaluate data mining independent datasets. Instead, they use information from the history of the creator to collect data. Due to the possible drawbacks of in-the-small data mining systems, they cannot be used when correlations must be created. Domain recommendation mechanisms allow

developers to reuse existing content components by finding out what they can use in a wide range of products. The recommender must remove the repository of goods, because RITS are inadequate for this business. Tiny consulting networks in rural areas are important and helpful.

RITs that do not use machine learning are not effective when there may not be a variety of software systems involved in a training set and when the developer attempts to use software for a software project not represented in a training data, which makes weak recommendations because of their reliance on static heuristics. A developer will begin development in the same way that a training kit does. However, as the project continues to become more complex, the guidelines will need to be reduced over time. All of these problems can be solved by adding or improving machine learning approaches with each new iteration.

To help their decision-making, RITSs do not use any data about non-localized trends. As the recommendations are based on the current context of the client, if the admin recommender has entrance to details regarding the remainder of the process, they may not be as successful as they should be.

As developers get started, they build a repertoire of knowledge they'll need to know. Not only does knowledge exist from a number of sources, but it can be conveniently obtained and integrated into the environment.

The project source code. The code base itself can overwhelm any software system with the amount of knowledge it possesses. Since the source code is only understood on a small scale, it needs to be interpreted to answer a number of different kinds of questions, such as "device’s name"[4]. We must browse the source code in [5,6] to address questions about the source code, such as reading comments, identifiers and dependencies.

The plans for the plant. Information about a software project is stored in a VCS, and the VCS captures all revisions. Useful knowledge includes regular code updates [7], the architecture of the code [8] and which programmers are aware of which section of the code [8]. Unfortunately, the VCS is not easy for users to search or navigate. Knowledge obtained by VCS and generated by other outlets, combined with intuition, heuristics, and data mining.

Database of Correspondence (lists), which are used by programmers to connect them with other teams, provide rich machine knowledge. In terms of communication, problems management and review applications are all registered.

The trusted software and learning resources. reusable programming objects (APIs) are used in a disproportionately large number of cycles after they're acquired. API's have a broad, well-structured framework that enables the operator to be able to perform their tasks. Further, in-depth documentation also includes comprehensive APIs, including reference guides, user manuals, and code samples.

The climate for development, opportunity. The software system needs programmers' skills, and its programming framework includes all of the programming facilities, instructions and commands used to build and validate the code. Developers can perform sub-optimally due to the tools and commands, thereby rendering any development environment problematic.

Proof of touch. Other than serving the customer, processing user interface data to improve usability is standard practice for business. User commitment data provide details of users' activities, particular actions taken, periods of use, and frequency of use. This software design research takes the form of tracking developers' behaviors when they use an IDE system, e.g., Eclipse.

Traces of authorship. Information obtained from systems is used by software developers as a source of information about quality of service. Data collection includes knowledge of the entire system state, its functions, and how the system acts under various inputs.

The Site any realization contained in the cloud can be correlated with the required development period. To solve programming problems, developers can go online and find code samples as an additional source of answers. The cloud problem is that it is always difficult to measure the accuracy of the data on several pages and it is often hard to predict the outcomes of search queries.

All knowledge in theory is available for further creation and other technical activity; it may be spiritually difficult to extract an answer to a specific demand for information, or even to know in certain instances that the response takes place. In computer science, a wide variety of software creation makes handling information in different ways particularly difficult.

We have used a dataset from the open-source project, where we use Naive Bayes, Decision Trees, Random Forest, and Neural Networks to generate a hybrid system. Several researchers have used the Vector Machines approach and have applied learning algorithms without legal constraints. Furthermore, they used open-source data and a Firefox project.

2. BACKGROUND

2.1 Background

The development of software remains a challenging and knowledge-heavy job as the technology keeps improving. In order to attain excellence, no longer is learning a programming language mandatory. New software, hardware and Web technologies are constantly developed. The networks they operate on tend to continue to grow with novel content generators and connections to numerous web services.

The traditional information space creator exceeds the natural capacity for absorption of an individual. Software developers can spend a large amount of time looking for details such as how to best use programming tools or how to use the code. The next discovery in this research will greatly affect the production results.

However, while thorough planning and good interpersonal communication can allow information workers to organize themselves, these techniques are restricted by scale. In addition to the more prevalent methods for providing automated system to programmers in managing large data, are intelligent approaches. As RS systems for popular e-learning platforms can lead consumers to fascinating things before time unrecognized [19], these systems might be utilized to help intelligence staff in their assignment.

software engineering recommendations systems (RSSEs) have been developed to assist in different ways, including code reuse, error handling, and more.

CF is easy to set up, while CBF is simple to set up. (only basic item-name and image-based information is needed), there are three unique forms of collaborative filtering for which three different types of scores are required, content-based filtering and knowledge-based recommendations leverage various forms of recommendation expertise which have varying strengths which disadvantage both the CF and the CBF. On the other side, utility systems have to be manually changed to incorporate the technology-based suggestions.

2.2 Al and RS

The recommendation is a term used frequently within the business world. Because RS algorithms are a company's own intellectual property, they are often not designed by using conventional solutions, but rather by using the company's specific circumstances. In light of this situation, there are a number of recommendation services that can be developed. Strands is a website which provides recommendations on how to use various information products and services. The aim of MyMediaLite is to create a tool for collaborative filtering systems. LensKit was developed to facilitate the coding and assessment stages of suggested solutions, especially collaborative filtering assessment algorithms MovieLens, a website that is close to YouTube, is an innovative concept. The MovieLens database is available for the purposes of study so that new algorithms can be checked. Apache Mahout is a machine-learning platform that provides recommendations to help minimize data bias. The restricted operating system libraries such as Choco and JaCoP allow for knowledge-based recommendation programs. WeeVis is a knowledge-based environment for recommendation-based applications. Recommendation applications can be deployed on different platforms including the Mac OS and the Windows.

Feedback and therapy services are becoming increasingly relevant because of the increased usage of the Internet.

Basic methodological concepts for guidance. Tourism (such as which location or tourism attractions) and television (such as a certain TV show) will be the implementation fields of community selection technologies. Public reviews are also focused on locations that have pleasant names, places with spectacular views, and places of interest.

The group is responsible for creating a strategy that can be decided upon by the entire collection. This is the heuristic judgment adopted by the group, which assigns the lowest vote to alternative treatments. Some ways of group recommendation such as the highest votes of the group are the happiest (the vote for a particular response depends on the number of individuals votes: the group is the top-value material). An exception should be made for community suggestions of things with high participation.

2.3 Software Development Cycles

SDLC is not only agile, but also iterative, and sequential. Agile methodologies, such as XP and Scrum, focus on lightweight processes which allow for rapid changes (without necessarily following the pattern of SDLC approach). Iterative methodologies such as Rational Unified Process and dynamic systems creation method benefit by restricting the complexity of a project so it can take more iterations. Sequential or big-design-up-front (BDUF) models, such as waterfall, focus on complete and accurate planning to direct large projects and risks to effective and predictable results. Other models, such as anamorphic development, tend to focus on a form of development that is driven by project scope and adaptive iterations of feature development.

In project management a project can be described in a number of different ways, including both a project life cycle (PLC) and an SDLC. According to J. Taylor (2004), "the project life cycle encompasses all the activities of the project, while the systems development life cycle focuses on realizing the product requirements".[5]

Systems Development Life Cycle (SDLC) is a mechanism by which the development of an IT system travels from the drawing board to the final completion.

The model is a common but not widely accepted method of describing the life cycle of a software application. These steps include study and analysis, design, development, testing, implementation, and maintenance. Both software development methodologies (for example, waterfall and scrum) follow the SDLC stages, but the method of doing so differs vastly between methodologies. In Scrum, a single user tale usually travels through all the stages of software development within a single two-week sprint. Contrast this to the waterfall approach, as another example, where any business requirement (recorded in the analysis phase of the SDLC in a document called the Business Requirements Specification) is converted into feature/functional descriptions (recorded in the design phase in a document called the Functional Specification) which are then all designed in one go as a series of solution features usually over a period of three to nine months, or more. Both methodologies are clearly very different methods, but they both include the SDLC phases in which a requirement is born, then moves through the life cycle phases culminating in the final phase of maintenance and support, after-which (typically) the entire life cycle begins again for a subsequent version of the software application.

Some approaches to recommendation systems for bug solving proposed by some scholars are briefly mentioned. In order to find the right developers for bug fixing in bug systems, several models and simulations have been built and suggested. In the problem reports of Eclipse and Mozilla problem repositories using the social network model [7], Xuan et al. suggested a model focusing on social awareness. An automated bug assignment using a time parameter was used by Shokri pour et al. Time metadata for the weighted expression was considered in the approach. The phrase repetition in records is set with the usage of a technique named tf-idf [20]. An enhanced Linear Discriminant model for bug assignment has been suggested by Xia et al. As a learning method named Topic Miner, they suggested a methodology with a gradual improvement, this Topic Miner technique defines the required problem reports for suitable developers with regard to the dissemination and resemblance of the issue. The similarity is generated as a partnership between programmer and delivery for bug fix [21].

This bug warehouses are indexed and referenced to a section of issues that produces a broad archive of all issues found in a project by software designers or programmers. Open bug archives are considered open source, through which these findings can be read by anyone. Thanks to the complete access consent of programmer groups to search, locate, distribute, modify, and patch throughout the project's creation progress, these archives play a significant function in such projects.

2.4 Machine Learning algorithms: the engine

The recommendation system usually consists of three blocks: the inputs, which are the data that required to be analyzed. The engine, which consists of machine learning or any artificial intelligence approach that process that data and display the results which they represented by the output block. For the engine block, It is recommended to use the most straightforward and simplest approach when conducting standardized testing of new data. When the baseline value has been defined, then by using the necessary analysis it is appropriate to consider the data.

The naive Bayesian method is the simplest, fastest approach by integrating a number of methods, such as learning from a variety of information sets, incremental learning, variance detecting, row pruning, and functional slicing, this process can be completed easily.

The critics bring up a new criticism of the Tree Analysis argument. As tree students collect data in contexts, they take advantage of algorithms such as C4.5 and CART which include all subtrees in their data. Strange, naive Bayesians frequently conduct a thoroughly documented experiment that is described as good or better than the experience of a decision-making phase in Domingos and Pazzani [22]. The area in which a Bayesian assumes the similitudes between something and what it looks like. If the number of characteristics in the dataset increases, the smaller the dataset is. This is an example of how very few rooms in modern Bayesian theories are simply naive. This is a fantastic news because it saves a lot of data on a single medium, thus:

• Low memory footprint; preparation takes a short time, and decisions are easy to take

It is easy to construct.

The Bayesian theorem is used by naive Bayesian classifiers to construct probabilistic predictions. Grades are separated into various time ranges. For each class, statistics are obtained. Review the game, for instance. To illustrate how this happened, take a look at the numbers. Note that the two dependent variables are measured on "yes" or "no" answers.

To play = Yes, to play = No Forecast temperature
Yes No
Clear 2 3
Cloudy 4 0
Wet 3 2

Clear 2/9 3/5
Cloudy 4/9 0/5
Wet 3/9 2/5

windy
==========
humidity
==========
yes no
86 85
96 90
80 70
yes no yes no
False part 6 2 9 5
True part 3 3 -------- ----------
False part 6/9, 2/5, 9/14, 5/14, True part 3/9, 3/5
yes no 83 85 70 80 68 65 ---------- 73 74.6 mean

79.1 86.2 10.2 9.7
mean standard dev. 6.2 7.9 standard dev.
Score: To Play.

Some accumulated statistics are included in each function. For example: For example:

• The total moisture content varies for the Yes part and No part of the law.

Cloudy happens when we play 4 out of 9 days, and not participate at all. In other words, if we see the expression 'forecast = cloudy', then we'll play more than average (play exponentially).

We have clustered all items together in the same cluster in the same way a naive Bayesian clustering can be viewed. When fresh data comes in, the cluster is split into two and those clusters that have the highest votes are used for the calculation.

The voting process utilizes Bayes' rule: the present confidence in a proposed method is the product of all the recent data from the days of the old faith, i.e., the latest degree is now existing. In the testing process, if a function is absent, we would assume that it does not hold any rationale for a decision. There is no position you can skip around. Concerning the use of the handling for missing values, several other courses are better than this one. In C4.5, it executes a complicated subtree; it operates on whether a variable is not found in the testing process.

"Probability of a hypothesis given evidence: Pr(H□E) = Pr(E □H) Ã Pr(H) / Pr" (E). In Bayes' law, the probability of a hypothesis given evidence E is defined as P(H|E).

• The Pr (E)= Probable Case, It could be skipped (because it is really vague), because all assessments were handled the same.

• The hypothesis is the hypothesis's previous probability (H). That is the rate at which each expectation occurs. In our case, H is Yes, and No, so the likelihood that H is Yes is 9/14, and the probability that H is No is 5/14.

• The likelihood of the given conclusively proved hypothesis, Pr(E□H), being returned from the list of statistical evidence.

As an example, if we were told today was going to be a clear day, the voting on the classes should be:

• Pr(yes succeeding-look = Clear) = 2/9 \times 9/14 = 0.39.

• Pr (Deciding to Stay) = 3/5 ÷ 5/14 = 0.14

As yes term is optimistic, we expect that tomorrow will be a good day. By understanding that the more knowledge that exists, the more important it becomes. Please see this example as well.

• Consider 67 degr. is reached. The forecast shows that the relative humidity will be 91%, e.g:"E= (Forecast=clear; temperature=67; humidity=91; windy=true).

• By increasing the likelihood of disjunctive terms: P(yes term E) = 2/9 x 0.0340 x 0.000036 = 0.000036.

• Pr (No Term) = 3/5 x 0.0291 x 0.0380 x 3/5 x 5/14 = 0.000136.

We have normalized possibilities and normalized them so that:

• (yes term="E") = 0.00036(0.00036 + 0:000136) = 21%

• (no term E) = 0.000136/(0.00036 + 0.000136)/(0.00036 + 0.000136) =79%

For all above, no more than one vote per notice is permitted. Given the above prediction, we'd expect no data.

There were big numbers in the formulas, such as 0.035. The experiment was conducted at a temperature of 67, and the average Yes response was 73 degrees Celsius. The standard deviation was 6.2. The chance of 67 coming up with an average of 73 and 62. We should explain that:

The closer to the mean value, the more probable the result would be.

As the collection of results increases, the potential for uniqueness goes down.

The notions described may all appear as a curve: Geometric distribution.

• Close to the average value, the graph reaches the limit.

• The curve gets wider as variance increases. The steeper the slope, the lesser the mean, as the sum of the elements under the chance curve will be one. Diversity curbs our belief in a specific trait.

Thus, for some classifier how to calculate the probability, a Gaussian distribution formula is used:.

Feature main()

The following graph:

{

return 1/(standard ev * sqrt(2*pi))* e^(-1 * (x - mean)^2/(2* standard ev * standard ev))

}

Print PDF with a Gaussian-based template.

End.}

Note that the underlying distribution is a "bell-shaped" Gaussian curve. While this is useful in engineering, it may not always be true. Before making predictions, some classifiers discretize the numbers into continuous values, thus removing the need for the Gaussian assumption. Recurrent result is that discretization reduces time and effort for those kinds of transformations.

2.5 J48 and C4.5

The learner of the C4.5 decision tree shows a clear proof of disregarding all but the minimal mix of components that lead to decisions. If C4.5 reads the raw golf details, it will probably focus on the game. The intent. What particular qualities would be included in the playful behavior to then be recorded? The following trees will be drawn as the report structure.

Forecast is clear.

• HUMIDITY <= 75: YES word.

• humidity greater than 75: no word, Forecast = overcast: yes term, Forecast = rain.

windy = real else rainy

• windy = False; else yes.

Bear in mind that the sub-trees are indented, and that a decision is made based on the colon (:). For starters, the top section of this tree states: "If the forecast is 75, we can play golf." Remember, the temple is not included in this decision tree, that is, the weather. This means that golf is influenced by cold and heat. Rather than this function, this other one is more significant.

C4.5 discusses the complexity of a function. Find the following table of five samples of non-spiel golf and nine examples of non-play golf. Notice that the simple distributions in the table for no and yes are 5/14 and 9/14 (respective). In the top, in the middle of the branch, a bright blue sky is

visible. The C4.5 method has now formed a branch in which all the rows in which the view is overcast is well. This subtree has the values of 0 and 100.

In structured settings, we advise our readers to look for data breaks to minimize the effect of data variability.

The variety is measured using the rate of standard deviation. In each golf course, the scores were 5/14 for Pr1 and 9/14 for Pr2. Next, E equals the entropy of ([5/6, 9/14]).

= -5/14 * log2(5/14) = 0.94 * log2(9/14).

for the sub-tree for chosen = cloudy.

By subtracting the zeroes and then adding nothing, then the answer is 4.

e = -1*log(2) + 0 = 0

Please note that there are two yeses and one no in a subtree with five rows. In other terms,

N2 = 5.

e2(2/5, 3/5) = 0.97.

There are five rows adjacent to rainy which can all be picked.

Three no, two yes. Therefore;.

N3 is 5.

edquo(3/5, 2/5) = 0.97

2.6 Multiple Decision Trees

The omnidirectional decision tree utilizes an omnidirectional decision tree architecture where unidirectional, multidirectional, or nonlinear multidirectional nodes may be present throughout the tree. The idea is that during development, a different model may be suitable for each judgment node corresponding to a particular subproblem discovered by the limited collection of training set

reaching the node and that the acceptable model should be found and used. Using the same type of nodes everywhere causes a general inductive bias to be presumed across the property. Selected nodes of different types are trained at each node in an omni-vary tree and contrasted using a statistical test set on a validity spectrum to determine which one become the better. In order to construct a more accurate one, a simpler one will be chosen. The results indicate that more interconnected nodes are used. The first level descriptors are necessary and acceptable as we step down the hierarchy. When we step closer to the leaves we have simpler problems, and at the same time, less data. Complex nodes are overfit in this situation and are rejected by statistical analysis. As we travel down the path, the number of nodes increases exponentially; therefore, most of the nodes are univariate, and the overall complexity level does not dramatically increase.

Decision trees are used to group data, but also for regression analysis. They can act as a fast learner and are especially concise. The decision tree is selected in a way that it can be interpreted. The D.tree can be seen as an automated enforcement framework that judges if you are abiding by the rules.

Generally, if the decision tree has a testing process and its accuracy is taken into account, more advanced methods such as factor analysis may be applied. Tree analysis enables comprehension of what trees are like and what they do. The univariate tree profits from being able to combine numerical and distinct characteristics without having to transform a kind into a different kind.

The Decision Tree model is to be compared to instance-based models and is yet to be tested.

Either node has a bin without the same size or of the same size as the others (such as the nearest k-nearest neighbor).

• The binned divisions use entropy or error-supervised output information on the basis of similarity in space.

• The advantage of the method is that with few measurements, we can distinguish leaf shapes much better

The decision tree only contains tree structure such as branches, nodes, and leaf values. This method is superior to non-parametric instance-driven approaches because it uses less room.

A class does not need to provide a clearly defined and sensible description with a decision tree for every particular case. It can have a number of different potential representations in the input space—including disjoint representations.

We refer to a decision tree we have so far. We take one of the divisions, depending on the exam. Starting from the root, we follow a single path until an answer value is reached at a given book. However, in the soft decision tree, we take all the roots at once, so we ensure that the probabilities are the same in all directions. The performance is the weighted average of all the outputs in all the projectors, with the weights depending on the probabilities applying to the path length.

As several learners may be involved, a decision tree is used as a mixed model for decision making, and a community of decision trees is called a forest. We will find that if we train one but several decision-making trees each with a random training subset or a random sub-set of inspection features, not decision-making forests and combining their predictions, overall accuracy can be improved greatly. This is the fundamental basis of the random forestry system.

2.7 Neural Networks

The artificial neural network model integrates the human brain and perception in order to predict actions. There are cognitive scientists and neuroscientists who research the brain and conduct computational experiments which aim to explain the actual workings of the brain.

Artificial neural networks architecture is of interest to us, and we hope that computer architectures will benefit from this study. In certain areas of the brain, for example, vision, speech comprehension, and listening, the brain is an information gathering machine that has abilities that transcend modern technology. When deployed to machines, these applications are important economically. We can find solutions to these problems by defining algorithms and performing them on machines to approximate the functions that the brain performs.

The human brain is very different from a computer. In comparison to a computer with a single processor, a brain may consist of multiple processors working in parallel. The processing units are believed to be artificial neural networks, but the details of this system are unconfirmed.

3. METHODOLOGIES

3.1 Eclipse data

The dataset was collected from the error tracker on eclipse.org. The kit consists of the 7700 data points of the result of the Eclipse platform bug measures dataset. In the first column of the table, the bug ID is mentioned. Last, the given columns contain the 48 team labels from 1 to 7. Each team of the dataset represents a group of developers according to their companies that assigned before to fix bugs in certain components of the software project. For these features, we tested several machine learning algorithms by using Matlab and Weka to classify features according to team labels and rely on features. It has been achieved based on the 10-fold cross-validation. The dataset is multidimensional. The next graph displays the number of developers involved, along with the number of bugs. Each group of developers is defined by an organization, and the number of bugs fixed by each team.

Table 3.1: Developers and instances in each team.

Team	**Number of developers and instances**	
	Number of developers	***Number of Instances***
1	10	1206
2	10	1173
3	19	735
4	60	844
5	11	1610
6	20	1280
7	8	918

Team	Number of developers and instances	
	Number of developers	***Number of Instances***
Total	138	7696

The teams of the dataset represent the different companies that are developing the app, as well as their total number. The Eclipse bug dataset can be found at https://github.com/logpai/bugrepo/tree/master/EclipsePlatform

3.1.1 Database Description

The columns of the Eclipse database consist of the following fields:

bugID; component;; assigneeEmail;os; platform; milestone; nrKeywords; nrDependentBugs; peopleCC;openedhoursOpenedBeforeNextRelease;lastModified;priority;severity;resolution;firstFix;lastFix;hoursLastFixBeforeNextRelease;hoursLastFixAfterPreviousRelease;status;firstActivity;nrActivities;lastResolution;nrComments;hoursToLastFix;hoursToLastResolution;monthOpened;yearOpened;monthYearOpened;monthYearLastFixed. These features represent the details and contents of each bug report. Below is an example of a bug report:

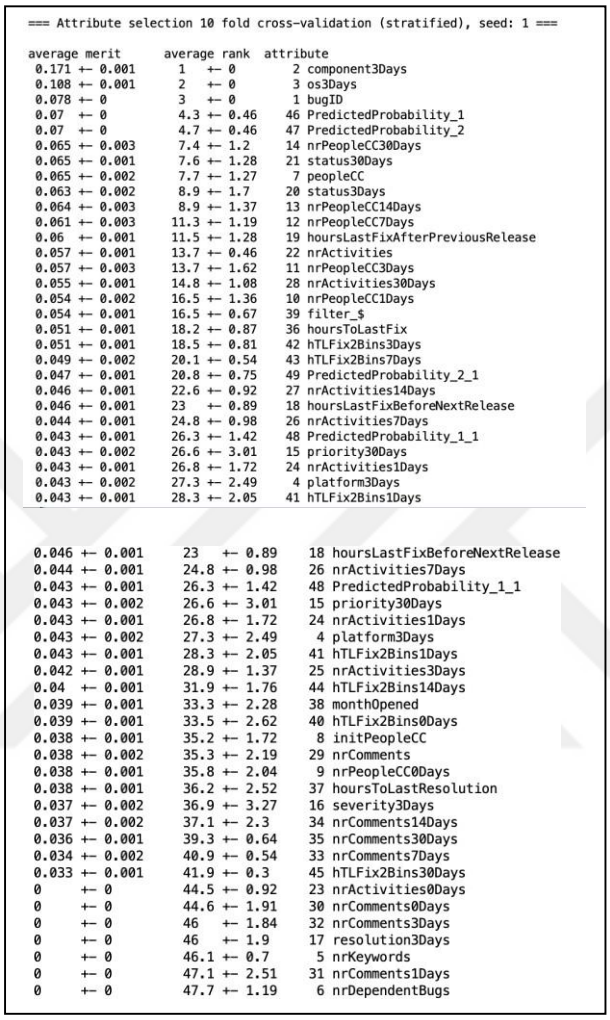
Figure 3.1: Bug report

3.2 Optimization results

The data set is used to train classifiers that can result in wrong classifications and may also take longer to train. The explanatory and clarifying features are redundant and contradictory and do not contribute to the classification. To boost classification accuracy, redundant and inconsistent features must be removed [1]. Use feature selection methods to classify combinations of features that maximize the amount of information that is collected. We omitted the final twenty features with the lowest correlation that could unduly influence our data set performance. Figure 2 displays the effects of the different features' correlations. These algorithms are constructed using natural selection dynamics and natural genetic mechanics. Genetic algorithms are used to solve string structures like biological structures, which develop in time by survival filtering using a randomized yet coordinated exchange of information. Thus, a new set of strings is created in each generation, with only the fittest individuals passing on from generation to generation. The Genetic Algorithm's fundamental characteristics are:

- The genetic algorithm operates with parameter set coding, not parameters themselves.
- The genetic algorithm begins looking from a point population, not a single point.
- The genetic algorithm uses payoff, not derivatives.
- Genetic algorithms use probabilistic, not deterministic, transformation laws.

A genetic algorithm is a type of stochastic algorithm that looks at probabilities. By applying a random search strategy to a step-by-step superstructure model, the search mechanism is determined. The optimum global solution can be reached at a chance of x% certainty. The search process is triggered by the selection of initial stochastic solutions called 'population.' These are called 'chromosomes.' 'Chromosomes' are made up of 'genes.' 'Gene' stands for the optimal variables in the heat exchanger network, such as the mass flow of cold streams and hot streams. For us, we used the GE algorithm to reduce the features used for classification by deleting those that don't have an impact on the final result.



```
=== Attribute selection 10 fold cross-validation (stratified), seed: 1 ===

average merit      average rank  attribute
 0.171 +- 0.001      1   +- 0        2 component3Days
 0.108 +- 0.001      2   +- 0        3 os3Days
 0.078 +- 0          3   +- 0        1 bugID
 0.07  +- 0          4.3 +- 0.46    46 PredictedProbability_1
 0.07  +- 0          4.7 +- 0.46    47 PredictedProbability_2
 0.065 +- 0.003      7.4 +- 1.2     14 nrPeopleCC30Days
 0.065 +- 0.001      7.6 +- 1.28    21 status30Days
 0.065 +- 0.002      7.7 +- 1.27     7 peopleCC
 0.063 +- 0.002      8.9 +- 1.7     20 status3Days
 0.064 +- 0.003      8.9 +- 1.37    13 nrPeopleCC14Days
 0.061 +- 0.003     11.3 +- 1.19    12 nrPeopleCC7Days
 0.06  +- 0.001     11.5 +- 1.28    19 hoursLastFixAfterPreviousRelease
 0.057 +- 0.001     13.7 +- 0.46    22 nrActivities
 0.057 +- 0.003     13.7 +- 1.62    11 nrPeopleCC3Days
 0.055 +- 0.001     14.8 +- 1.08    28 nrActivities30Days
 0.054 +- 0.002     16.5 +- 1.36    10 nrPeopleCC1Days
 0.054 +- 0.001     16.5 +- 0.67    39 filter_$
 0.051 +- 0.001     18.2 +- 0.87    36 hoursToLastFix
 0.051 +- 0.001     18.5 +- 0.81    42 hTLFix2Bins3Days
 0.049 +- 0.002     20.1 +- 0.54    43 hTLFix2Bins7Days
 0.047 +- 0.001     20.8 +- 0.75    49 PredictedProbability_2_1
 0.046 +- 0.001     22.6 +- 0.92    27 nrActivities14Days
 0.046 +- 0.001     23   +- 0.89    18 hoursLastFixBeforeNextRelease
 0.044 +- 0.001     24.8 +- 0.98    26 nrActivities7Days
 0.043 +- 0.001     26.3 +- 1.42    48 PredictedProbability_1_1
 0.043 +- 0.002     26.6 +- 3.01    15 priority30Days
 0.043 +- 0.001     26.8 +- 1.72    24 nrActivities1Days
 0.043 +- 0.002     27.3 +- 2.49     4 platform3Days
 0.043 +- 0.001     28.3 +- 2.05    41 hTLFix2Bins1Days


 0.046 +- 0.001     23   +- 0.89    18 hoursLastFixBeforeNextRelease
 0.044 +- 0.001     24.8 +- 0.98    26 nrActivities7Days
 0.043 +- 0.001     26.3 +- 1.42    48 PredictedProbability_1_1
 0.043 +- 0.002     26.6 +- 3.01    15 priority30Days
 0.043 +- 0.001     26.8 +- 1.72    24 nrActivities1Days
 0.043 +- 0.002     27.3 +- 2.49     4 platform3Days
 0.043 +- 0.001     28.3 +- 2.05    41 hTLFix2Bins1Days
 0.042 +- 0.001     28.9 +- 1.37    25 nrActivities3Days
 0.04  +- 0.001     31.9 +- 1.76    44 hTLFix2Bins14Days
 0.039 +- 0.001     33.3 +- 2.28    38 monthOpened
 0.039 +- 0.001     33.5 +- 2.62    40 hTLFix2Bins0Days
 0.038 +- 0.001     35.2 +- 1.72     8 initPeopleCC
 0.038 +- 0.002     35.3 +- 2.19    29 nrComments
 0.038 +- 0.001     35.8 +- 2.04     9 nrPeopleCC0Days
 0.038 +- 0.001     36.2 +- 2.52    37 hoursToLastResolution
 0.037 +- 0.002     36.9 +- 3.27    16 severity3Days
 0.037 +- 0.002     37.1 +- 2.3     34 nrComments14Days
 0.036 +- 0.001     39.3 +- 0.64    35 nrComments30Days
 0.034 +- 0.002     40.9 +- 0.54    33 nrComments7Days
 0.033 +- 0.001     41.9 +- 0.3     45 hTLFix2Bins30Days
 0     +- 0         44.5 +- 0.92    23 nrActivities0Days
 0     +- 0         44.6 +- 1.91    30 nrComments0Days
 0     +- 0         46   +- 1.84    32 nrComments3Days
 0     +- 0         46   +- 1.9     17 resolution3Days
 0     +- 0         46.1 +- 0.7      5 nrKeywords
 0     +- 0         47.1 +- 2.51    31 nrComments1Days
 0     +- 0         47.7 +- 1.19     6 nrDependentBugs
```

Figure 3.2 : Attribute Correlation ranker

3.3 Methodology configuration

Machine learning algorithms like Naive Bayes, Decision Trees, and SimpleLogistic will be used for making the recommendation engine, and unsupervised machine learning algorithms like Expectation Maximization might also be used, [24]. First, we have tried hybrid of two machine learning techniques with optimization of genetic algorithm in order to create a recommendation engine. Feature selection algorithms will be used to get the best features that distinguish the two groups. We have tried optimization algorithm to choose the best features and implemented the Naive Bayes algorithm [25], I used the Naive Bayes algorithm for classification of the data set

using MATLAB code [26] and WEKA program [27]. As it is seen from the next figure 3.3, the flowchart of the system had been explained.

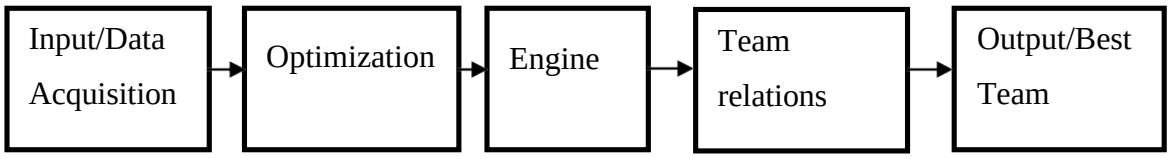


Figure 3.3: System Flowchart

3.3.1 Naive Bayes

Below, are the theorems used to allow the classification of the dataset in Bayesian probability for more than two groups [25].

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{\sum_{k=1}^{K} p(\mathbf{x} \mid C_k)P(C_k)} \tag{8}$$

$$P(C_i) \geq 0 \text{ and } \sum_{i=1}^{K} P(C_i) = 1$$
$$\text{choose } C_i \text{ if } P(C_i \mid \mathbf{x}) = \max_k P(C_k \mid \mathbf{x}) \tag{9}$$

- *Decision Tree*

For each node m, N_m samples can be reached from m, N_m via $Class_i$.: [25]

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p_m^i = \frac{N_m^i}{N_m} \tag{10}$$

If p = 0 or p = 1 then m is said to be 'pure' we can then measure the impurity with which it is entropy.: [25]

$$I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i \tag{11}$$

Build a leaf and stop if the leaf is pure. Otherwise, continue splitting recursively. Measure the impurity ratio that is related to branch j. N^i_{mj}taken from to C .$_i$

Finding the main concepts and dividing up the minimum divisor (among all variables and splitting positions for numeric variables) [25].

We also have applied the decision tree algorithm to the filtered dataset using the WEKA program with 0.25 confidence factor is utilized for pruning that lower values occur more pruning and 2 minNumObj is minimum instances per leaf and 3 numFolds that defines the amount of data utilized for minimizing the error in pruning. Only one-fold is used for pruning, while the others are in use for tree development.

3.3.2 Tree configurations

For node *m*, N_m examples in *m* domain, N^i_m for Class$_i$*:* [25]

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m} \tag{16}$$

If p^i_m is equal to zero or one, then Node *m* is called pure and we mathimatcally calcualte the impurity in which it is entropy: [25]

$$\mathrm{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m \tag{17}$$

Create a leaf and stop if node *m* is pure, otherwise continue split recursively. Calculate the Impurity after split: N_{mj} from N_m taking branch *j*. N^i_{mj}taken from to C_i.

Finding the variables and splitting that minimum impurity (among all variables and splitting positions for numeric variables) [25].

3.3.3. Multiple Decision Trees

This is an ensemble of one-by-one trained models which receives a subset of dataset observations and a subset of variables as branches of its tree to construct multiple decision trees (neural networks, support vector machines, decision trees as an example). In general, it provides high

classification accuracy as well as good generalization efficiency over various datasets. The algorithm combines multiple decision trees and seeks to improve accuracy by combining them and avoiding in-fit and out-of-fit issues. The prediction is based on the average of the voting of multiple judges, which should be better than any one judge.

If a model is to be built, it is constructed in a manner that the models are generated one after another. To train a classifier, we change the weights of all training instances depending on the accuracy of the classifier we are attempting to train. The model classifiers generated by using artificial intelligence are also unstable [29].

A new sample is analyzed by running it against the trees which produced the forest. Every tree has some initial classification score which will affect how a new sample will be rated. In a majority vote, the results from all trees are combined and the class in which the largest number of votes is counted is considered the classification outcome of this new survey. As suggested by Breiman [30], these random forests use majority voting to predict categorical responses. All experiments performed with voting.

We used the random forest classifier. We hope the findings would be beneficial with respect to those posts. The random forest consists of hundreds of tree classifier models. , hutK, huK (X), K for tree number. Each classifier model in the system is tested using a combination of training data and customer feedback. A test sample is identified by class label of the majority voting, as it is called. The criteria that we can use the Random Forest are:

The tree's deepest point, no limit the maximum depth, = 0.

numFeatures = 0 if numFeatures < 1 input is logR + 1 if numFeatures > 1 input is logR + M We also tested many different features like 10 or 20 but nothing seemed to make a difference.

number of trees to be built = 150

We have developed a model of Random Forest using 150 trees, each designed using five characteristics with out of bag error of 0.3. The Out of Bag is explained as follows: after implementing Random forest model that is five trees, for each, x_i,y_i dimensions in the training set T_s, selecting all T without including x_i,y_i. This small collection is a set of boostrap samples which does not include a particular register from T_s dataset.

These sets of data are subsets of the training dataset, so we call them N subsets. Out-of-bag model classifier is the merge of votes from only T sets so that individual votes for element x_i, y_i are not included. For the out-of-bag estimations error, the error ratio of the out-of-bag model on the training set as opposed to the known class labels.

The objective of the study of error speculates that bagged model classifiers have high accuracy when utilized against an untrained testing dataset of the same size as the training dataset. By using OOBE we can remove the need for test data altogether.

3.3.4 Neural Networks with optimization

The network-based approach was created by the biological network model that describes how the brain functions. This method learns by analyzing the data and making predictions based off that data, without any additional laws. An example of image recognition where it learns to detect images that contain computers by training some image samples labeled manually as "computer" or "not computer" and by using the approach, it can output to detect cars in other unrelated images. The method can be used without prior familiarity; it automatically calculates and distinguishes important characteristics from the entered data.

The neural networks have linked nodes which represents the neurons in the human brain. Each synapse in a biological brain is responsible for transmitting signals to other brain cells (nodes). The same ideas are used by the receiving signal, processing it, and transmitting a signal to other related nodes.

In a bio-neural network, a signal is interpreted as an artificial neural network value. The output of the nodes is the product of the processing of the total inputs of the node. The nodes are interconnected through lines called edges. Nodes and edges are given weights throughout the learning stage that are modified during the learning process. When the weight increases, the intensity of the connection decreases, and the connection is enabled at the threshold, which is given by the weight. Nodes are usually composed of several layers.

Each layer contains many nodes that perform many different kinds of computations on their inputs. A signal transmits from the first input layer which is known as the input layer to the final output layer which is known as the output layer after receiving input from the hidden layers several times

depending on the threshold and accuracy of the best performance. We have investigated the architecture of neural networks as a training model with the hybrid method of decision tree and naïve Bayes to decide the best output for the assignment of open-source system bug reports to the appropriate developer. The genetic optimization algorithm is used with machine learning approaches to select the best teams among other teams, it works as a filter to the engine phase.

3.4 Team Relations database

- Computing required time for each programmer in the training and testing sets and determine the demand of each programmer in bug's class by utilizing the following:
- Determine the time required for each programmer in group.
- Find the differences in time.
- Arrange the programmers: the faster programmer have the highest rank.

4. FINDINGS

4.1 Optimization Engine outputs

Our methodology acquired precision rates higher than 60 percent and we assumed that the precision rates they registered are adequate to enable the bug triager to determine which developers are good enough to be assigned to a particular bug report [3]. We have used 10-fold cross validation and attribute correlation to apply features and interpret data. In the development of our bug measures, we utilized Naive Bayes, J48, Simplelogstic, random tree, and artificial neural networks. By using an Intrinsic Network, we've obtained the best outcomes.

We also found that there are features not appropriate for the classification and made the results even worse and others gave higher rate to the classification accuracy. We have optimized for accuracy by taking the best thirty features which have high attribute correlation values.

Table I displays the model output of various machine learning algorithms like Naïve Bayes, random trees, Simple Logistic and Artificial Neural Networks with Learning rate: 0.1, momentum: 0.55, batchsize: 105, and 200 to 500 iterations.

Regarding the number of trees in the random forest, the results obtained suggest that often, a greater number of trees in a forest just raises its computational cost and has no meaningful output advantage and this is what happened in the dataset. When we utilized more than 150 decision trees. we tried 600 decision trees, but no significant changes are observed. We have tried to find a way to boost the classification accuracy and the efficiency of the algorithms which is why we used optimization. The modification of class labels in the clustering algorithm created better outcomes. Meanwhile the demand of each programmer in the same class is determined by utilization of the last hours' features and the results are shown in table 3. Table 1 shows t he effects of optimization and hybrid use of Naïve Bayes algorithm and ANN together in the training set and the utilization of J48 with Simple logistic (50 percent -50 percent) in the training set, we can declare it has shown better results than the usage of each one alone approximately

7.42 percent improved results in optimization, 8.21 percent for hybrid optimization performance.

Table 2 shows the results of different machine learning algorithm I used like Naïve Bayes, Decision Trees, Random Forest, Simple Logistic and Neural Network (Learning rate: 0.3, batchsize:100.

Table 4.1: Accuracy findings

Algorithm	***Classification accuracy (%)***
J48 Trees	61.24
Random Trees	51.39
SimpleLogistic	61.27
Naïve Bayes	49.08
Artificial Neural Networks	63.12
J48 Trees with Optimization	65.59
Random Trees with Optimization	55.21
SimpleLogistic with Optimization	65.82
Naïve Bayes with Optimization	52.72
Artificial Neural Networks with Optimization	67.81
Hybrid O-J48 with O-SimpleLogistic	66.59
Hybrid O-ANN with O-J48	67.59
Hybrid O-Naïve Bayes with O-ANN	60.98
Hybrid O-ANN with Random Trees	62.24

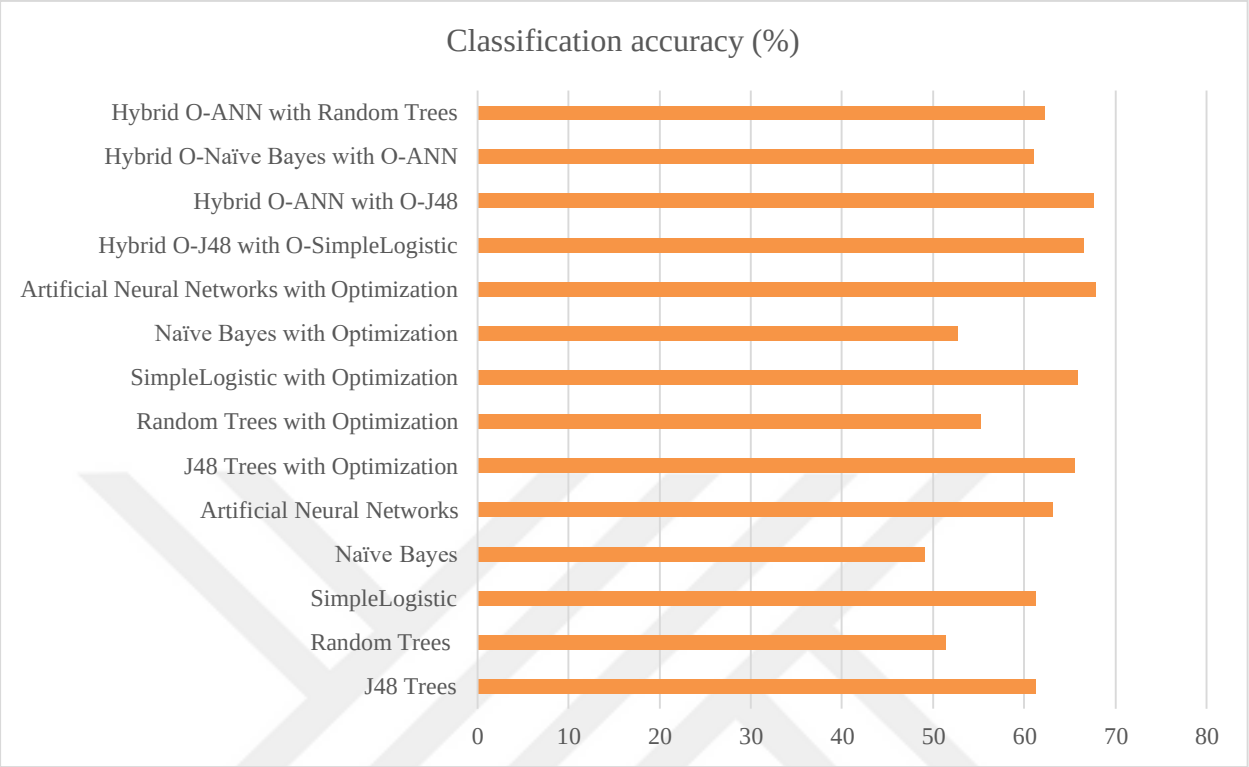


Figure 4.1: Classification scores

In the random forest approach the accuracy results acquired have shown occasionally that as the number of trees increases in a forest only increases its computational time, and expense, and has no considerable changes in performance benefit and this is what occurred in our datasets when we have used more than 150 trees. We have tested 600 trees, 5,000 trees, and nothing we've tried gets results. So, we have tried several methods to improve the classification accuracy and the efficiency of the algorithms, and we have found that we can make some improvements to the class labels including normalization and clustering as it is shown in the next section (clustering results). For developer priority, we measured the hours each developer had worked in the same project using the last hours' feature, and the accuracy results are shown in appendix B.

4.2 Optimization with Clustering scores

The system efficiency can be enhanced by modifying the programmer's classes or sets using a K-means algorithm with K=1000. We will have seven clusters (classes) with the closest class for each programmer, the programmer with the closest mean value will belong to that class. In table 2, the examples of programmers and the clusters they belong to are shown.

Table 4.2: Instances in each new team

New Team	***Samples***
1	*1572*
2	*1223*
3	*1180*
4	*1088*
5	*948*
6	*901*
7	*854*

After applying the new classes, we have utilized the same algorithms with 10-fold cross validation, and we have made better results (approx. 12.1% improvement).

5. CONCLUSION AND FUTURE ASPECTS

The proposed framework methodology would assist the bug triage programmers in administering and selecting which will be a functioning utility to fix a certain form of bug reports by using the open-source bugs repository. For assignment of bug reports. We used machine learning statistical approaches such as Naïve Bayes, J48, random forests, and Simplelogsitic.

For the feature collection, we have used information benefit values and genetic optimization to decide which features make decisions and we have excluded the 20 least insightful ones based on the most important 30 features.

Results obtained show that we have chosen to identify bug reports using algorithms such as random forest and neural networks. To support my conclusion, several research papers seem to indicate [7]. It is more accurate than both conventional decision trees and help vector machines for most classifications. Two advantages that should inspire us to choose random forest. One major benefit of using this model is that it does not require interactive features. Tree Ensembles combine the strength of Decision Trees, which function well for large datasets. The other key benefit of deep learning is that they are well-suited for solving high-dimensional problems and manage vast quantities of data well.

For the parameters of the random forest, we can discover that the good number of trees for the random forest is 150 trees, the accuracy for a random forest depends on the strength of the individual tree classifiers, and a measure of the dependency between them. The trees are categorized by the rating system that earns the most votes. Often, by increasing the number of trees in the random forest could only increases the processin g computational time while has no fair change in per- second output earnings.

Utilizing hybrid and optimization approaches, these machine learning algorithms provided an efficient bug assignment framework that meets the requirements. For the feature selection, we have used the information gain values to detect which features are good for

classification and we have eliminated 20 features because they had low information gain. So, we will do the classification based on the best 30 features and one feature for each class label.

The classification results have shown why we have selected algorithms such as random forest and neural networks to classify the bug reports. To support my conclusion, many research papers are approved that suggestion [28]. In general, it will give more accuracy than traditional decision tree and Support vector machines. Two advantages that let us select random forest for our dataset. First advantage is that they do not anticipate linear features or linearly interacted features. The branch of the tree is just a decision point of entire trees combined; this can deal with the dataset perfectly. The other advantage is that, because of how they are constructed, this approach deals perfectly with more than two dimensional spaces as well as large datasets of training samples [34].

For Multiple decision tree parameters, we have found that the good number of random trees to use is approximately 150 trees, and that the classification accuracy of a random forest relies on durability of the individual tree model classifiers and dependency between trees. The trees are categorized by how they are classified by the majority voting system. Often, increasing the number of trees in the random forest technique is ineffective at improving model efficiency. The hybrid method of using random forest and neural networks for the classification process and decision trees and naïve Bayes for the anomaly detection. Future research can use only such neutral-term-selection filtering algorithms in order to focus on the words most important for bug reports. Chi-square analysis is sufficient for the statistical power of the experiment. It's important to incorporate a new algorithm. We should consider a number of different methods instead of just this one.

The time efficiency of the framework is calculated in efficiency. Our approach analyzes the relationship between the characteristics of an ideal developer and the severity of bug reports. Present developers do not need to retrain the predictive models as new data arrives. This strategy lessens how long it takes for the framework to be up to date. But use the system's awareness for each specific period to maintain the efficacy of the system.

Many potential improvements and successful solutions can be considered, taking into account that is always necessary that system should work automatically for evaluating, recognizing and appropriate feature selection, for instance code testing and time while executing different tasks for adaptation and resolution of incoming new tasks automatically.

REFERENCES

[1] B. Azhagusundari, A. S. Thanamani, “Feature Selection based on Information Gain”. *IJITEE*, Vol.2, no.2, 2013.

[2] A. Goyal, N. Sardana, “Empirical Analysis of Ensemble Machine Learning Techniques for Bug Triaging”, *IC3*, IEEE, 2019.

[3] E. Alpaydin, “Introduction to Machine Learning”, 3nd edition, MIT press, London, England, 2014.

[4] B.Dagenais, H. Ossher, R.K. Bellamy, M.P. Robillard, “Moving into a new software project landscape”, *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pp. 275–284, 2010.

[5] A.J. Ko, B.A. Myers, M.J. Coblenz, H.H. Aung, “An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks”, *IEEE Trans. Software Engineering*, vol. 32, no.12, pp. 971–987, 2006.

[6] M.P. Robillard, W. Coelho, G.C. Murphy, “How effective developers investigate source code: An exploratory study”, *IEEE Trans. Software Eng*., vol. 30, no. 12, pp. 889–903, 2004.

[7] T. Zimmermann, P. Weißgerber, S. Diehl, A. Zeller, “Mining version histories to guide software changes”, *IEEE Trans. Software Eng.*, vol.31, no.6, pp. 429–445, 2005.

[8] A. Mockus, J.D. Herbsleb, “Expertise Browser: A quantitative approach to identifying expertise”, *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pp. 503–512, 2002.

[9] T. Zimmermann, P. Weißgerber,”, Preprocessing CVS data for fine-grained analysis”, *Proceedings of the International Workshop on Mining Software Repositories*, pp.2–6, 2004.

[10] D. Cubranic, G.C Murphy, J. Singer, K.S. Booth, “Hipikat: A project memory for software development”, *IEEE Trans. Software Eng*., vol.31, no.6, pp. 446–465, 2005.

[11] B. Dagenais, M.P. Robillard, “Creating and evolving developer documentation: Understanding the decisions of open-source contributors”, Proceedings *of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 127–136, 2010.

[12] E. Murphy-Hill, R. Jiresal, G.C. Murphy, “Improving software developers’ fluency by recommending development environment commands”, *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 1–11, 2012.

[13] W.C. Hill, J.D. Hollan, D.A. Wroblewski, T. McCandless, “Edit wear and read wear”, *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 3–9 ,1992.

[14] M. Kersten, G.C. Murphy, “Mylar: A degree-of-interest model for IDEs”, *Proceedings of the International Conference on Aspect-Oriented Software Development*, pp. 159–168, 2005.

[15] M.P. Robillard, E. Bodden, D. Kawrykow, M. Mezini, T. Ratchford, “Automated API property inference techniques”, *IEEE Trans. Software Eng*., vol.39, no. 5, pp. 613–637, 2013.

[16] J. Brandt, P.J. Guo, J. Lewenstein, M. Dontcheva, S.R. Klemmer, “Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code”, *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 1589–1598, 2009.

[17] O. Kononenko, D. Dietrich, R. Sharma, R. Holmes, “Automatically locating relevant programming help online”, *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 127–134, 2012.

[18] E. Duala-Ekoko, M.P. Robillard, “Asking and answering questions about unfamiliar APIs: An exploratory study”, *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pp. 266–276, 2012.

[19] P. Domingos, M.J. Pazzani, “On the optimality of the simple Bayesian classifier under zero-one loss”, *Mach. Learn*, vol.29, no. 2–3, pp. 103–130, 1997.

[20] S. Garcia, J. Derrac, J.R. Cano, F. Herrera, “Prototype selection for nearest neighbor classification: Taxonomy and empirical study”, *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 34, no.3, pp. 417–435, 2012.

[21] C. Gupta, R. Grossman, “GenIc: A single pass generalized incremental algorithm for clustering”, *Proceedings of the SIAM International Conference on Data Mining*, pp. 147–153, 2004.

[22] P. Domingos, M.J. Pazzani, “On the optimality of the simple Bayesian classifier under zero-one loss”, *Mach. Learn.*, vol. 29, no.2–3, pp. 103–130, 1997.

[23] J.R. Quinlan, “C4.5: Programs for Machine Learning”, Morgan Kaufmann, San Francisco, 1993.

[24] D. Marr,*Vision.*, New York, Freeman, 1982.

[25] A. Christopher Choquette-Choo, Sheldon D., J. Proppe, J. A. Gibbs, H. Gupta, “A multi-label, dual-output deep neural network for automated bug triaging”, *Intel PSG*, San Jose, California, United States, IEEE, 2019.

[26] MATLAB Documentation, http://www.mathworks.com/help/matlab.

[27] WEKA Manual, version 3-6-2, University of Waikato, New Zealand, 2020.

[28] J. Anvik, L. Hiewand, G. Murphy, “Who Should Fix this Bug,” *ICSE*, Shanghai,China, May 20-28, 2006.

[29] V. Y. Kulkarni, P. K. Sinha, “Random Forest Classifiers: A Survey and Future Research Directions”, *International Journal of Advanced Computing*, vol.36, no.1, 2013.

[30] M. Liua, C. M. Wangb, J. Wanga, D. Lic, “Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and Chinese vinegar”, *Sensors and Actuators*, vol. 177, pp. 970–980, 2013.

[32] Y. Ma, L. Guo, B. Cukic, “A Statistical Framework for the Prediction of Fault-Proneness”, West Virginia University, Morgantown.

[33] L. Breiman, OUT-OF-BAG ESTIMATION, Statistics Department University of California, 1996.

[34] Y. Zhao, Y. Zhang, “Comparison of decision tree methods for finding active objects”, National Astronomical Observatories, 2007.

[35] T. M. Oshiro, P. S. Perez, J. A. Baranauskas, “How Many Trees in a Random Forest”, Department of Computer Science and Mathematics, University of Sao Paulo, Lecture Notes in Computer Science 07, 2012.

[36] J. Romero-Mariona, J., H. Ziv, D.J. Richardson, “SRRS: A recommendation system for security requirements”, *Proceedings of the International Workshop on Recommendation Systems for Software Engineering,* 2008.

[37] F. Palma, H. Farzin, Y.G. Guéhéneuc, N. Moha, “Recommendation system for design patterns in software development: An [sic] DPR overview”, *Proceedings of the International Workshop on Recommendation Systems for Software Engineering,* 2012.

[38] M. Bruch, M. Monperrus, M. Mezini, “Learning from examples to improve code completion systems”, *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 213–222, 2009.

[39] P.N. Tan, M. Steinbach, V. Kumar, “Introduction to Data Mining”, Addison-Wesley, Reading, MA, 2005.

[40] G. Hikai, C. Shifei, S. Wang, D. Zhang, Y. Liu, C. Guo, H. Li, T. Li, “A Multi-Factor Approach for Selection of Developers to Fix Bugs in a Program”, *Applied sciences,* vol.9, pp. 3327, 2019.

[41] J. Xuan, H. Jiang, Z. Ren, W. Zou, “Developer Prioritization in Bug Repositories”, *IEEE,* 2012.

[42] J. Xuan, H. Jiang, Z. Ren, W. Zou, “Developer prioritization in bug repositories”, *Proceedings of the 2012 34th International Conference on Software Engineering (ICSE)*, Zurich, Switzerland, pp. 25–35, 2-9 June 2012.

[43] W. Wu, W. Zhang, Y. Yang, Q. Wang, “Time series analysis for bug number prediction”, *2nd International Conference on Software Engineering and Data Mining*, Chengdu, China, 23-25 June 2010.

[44] Y. Liu, X. Wang, Z. Zhai, R. Chen, B. Zhang, Y. Jiang, “Timely daily activity recognition from headmost sensor events”, *ISA Trans.,* 2019.

[45] W. eng, J. Xu, H. Zhao, “An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem”, *IEEE Access*, vol.7, pp. 20281–20292, 2019.

[46] J. Anvik, “Evaluating an Assistant for Creating Bug Report Assignment Recommenders”, June 2019, [online] Available: http: //ceur-ws.org/Vol-1705/04-paper.pdf, 2019.

[47] S. Guo, R. Chen, H. Li, T. Zhang, Y. Liu, “Identify Severity Bug Report with Distribution Imbalance by CR-SMOTE and ELM”, *Int. J. Softw. Eng. Knowl. Eng.,* vol. 29, pp.139–175, 2019.

[48] I. Aljarah, S. Banitaan, S. Abufardeh, W. Jin, S. Salem, “Selecting Discriminating Terms for Bug Assignment: A Formal Analysis”, *PROMISE*, Banff, Canada, 20-21 September, 2011.

APPENDIX

Appendix A: Solved bugs for each developer

CLASS NUMBER - Emails of developers	C	No of solved bugs
CLASS 1		
steve_northover@ca.ibm.com	1	301
john_arthorne@ca.ibm.com	1	266
James_Moody@oti.com	1	175
James_Moody@ca.ibm.com	1	139
Steve_Northover@oti.com	1	134
John_Arthorne@oti.com	1	78
knut_radloff@us.ibm.com	1	46
Knut_Radloff@oti.com	1	36
Lynne_Kues@oti.com2	1	20
jeff-bugs@code9.com	1	11
CLASS 2		
Kevin_McGuire@ca.ibm.com	2	273
dj_houghton@ca.ibm.com	2	201
jean-michel_lemieux@ca.ibm.com	2	192
Jean-Michel_Lemieux@oti.com	2	144
Kevin_McGuire@oti.com	2	124

Mike_Wilson@oti.com	2	71
Mike_Wilson@ca.ibm.com	2	57
DJ_Houghton@oti.com	2	56
debbie_wilson@ca.ibm.com	2	35
Debbie_Wilson@oti.com	2	20
CLASS 3		
christophe_cornu@ca.ibm.com	3	175
andre_weinand@ch.ibm.com	3	105
Boris_Shingarov@oti.com	3	105
Andre_Weinand@oti.com	3	101
Christophe_Cornu@oti.com	3	66
Carolyn_MacLeod@ca.ibm.com	3	65
Kevin_Haaland@oti.com	3	24
Carolyn_MacLeod@oti.com	3	23
lynne_kues@us.ibm.com	3	21
lynne_kues2@us.ibm.com	3	13
Jeff_McAffer@oti.com	3	13
erich_gamma@ch.ibm.com	3	6
Eduardo_Pereira@oti.com	3	5
Kai-Uwe_Maetzel@oti.com	3	5
Rodrigo_Peretti@oti.com	3	4

Kevin_Haaland@ca.ibm.com	3	1
jeff_mcaffer@ca.ibm.com	3	1
rodrigo_peretti@ca.ibm.com	3	1
rodrigo_peretti2@ca.ibm.com	3	1
CLASS 4		
gheorghe@ca.ibm.com	4	194
felipe_heidrich@ca.ibm.com	4	187
Felipe_Heidrich@oti.com	4	62
eclipse.rc@gmail.com	4	45
tjwatson@us.ibm.com	4	40
Rafael_Chaves@ca.ibm.com	4	38
silvio_boehler@ca.ibm.com	4	20
bshingar@ca.ibm.com	4	18
ymnk@jcraft.com	4	18
zhiyongl@ca.ibm.com	4	17
pascal_rapicault@ca.ibm.com	4	16
kcornell@rational.com	4	15
Erich_Gamma@oti.com	4	12
jed.anderson@genuitec.com	4	12
dean_roberts@ca.ibm.com	4	12
jeff_brown@oti.com	4	11

Dirk_Baeumer@oti.com	4	11
Daniel_Megert@oti.com	4	9
Nick_Edgar@oti.com	4	8
Rafael_Chaves@oti.com	4	7
Simon_Arsenault@oti.com	4	5
Tim_Ellison@uk.ibm.com	4	5
curtis_d'entremont@oti.com	4	5
Tod_Creasey@oti.com	4	4
dejan@ca.ibm.com	4	4
Antonio_D'souza@oti.com	4	4
cdm@qnx.com	4	4
Sebastian_Peleato@ca.ibm.com	4	4
equinox-inbox@eclipse.org	4	4
Shantha_Ramachandran@oti.com	4	3
Tod_Creasey@ca.ibm.com	4	3
klicnik@ca.ibm.com	4	3
sridhar@ics.com	4	3
Jed_Anderson@oti.com	4	3
rchaves.eclipse@gmail.com	4	3
lkues@us.ibm.com	4	3
lindaw@ca.ibm.com	4	3

mvm@ca.ibm.com	4	3
simon_arsenault@ca.ibm.com	4	2
Jim_des_Rivieres@oti.com	4	2
Adam_Kiezun@oti.com	4	2
jdt-core-inbox@eclipse.org	4	2
pde-ui-inbox@eclipse.org	4	2
nick_edgar@ca.ibm.com	4	1
kcornell@ca.ibm.com	4	1
birsan@ca.ibm.com	4	1
Olivier_Thomann@oti.com	4	1
johnrose@austin.ibm.com	4	1
Jed_Anderson@us.ibm.com	4	1
Chris_McLaren@oti.com	4	1
chris_mclaren@ca.ibm.com	4	1
jdt-ui-inbox@eclipse.org	4	1
Charlie_Gracie@oti.com	4	1
Darin_Swanson@oti.com	4	1
Philippe_Mulet@oti.com	4	1
max_sigalov@ca.ibm.com	4	1
mike_mallett@ca.ibm.com	4	1
douglas.pollock@magma.ca	4	1

pde-doc-inbox@eclipse.org	4	1
wassimm@ca.ibm.com	4	1
CLASS 5		
platform-team-inbox@eclipse.or	5	428
platform-cvs-inbox@eclipse.org	5	398
platform-vcm-inbox@eclipse.org	5	222
platform-resources-inbox@eclip	5	152
platform-swt-inbox@eclipse.org	5	141
platform-runtime-inbox@eclipse	5	134
platform-compare-inbox@eclipse	5	88
platform-core-inbox@eclipse.or	5	37
platform-text-inbox@eclipse.or	5	5
platform-doc-inbox@eclipse.org	5	4
Platform-ui-Inbox@eclipse.org	5	2
CLASS 6		
Silenio_Quarti@ca.ibm.com	6	300
grant_gayed@ca.ibm.com	6	294
veronika_irvine@ca.ibm.com	6	214
Silenio_Quarti@oti.com	6	166
Veronika_Irvine@oti.com	6	106
Grant_Gayed@oti.com	6	71

krbarnes@ca.ibm.com	6	47
billy.biggs@eclipse.org	6	43
bbiggs@ca.ibm.com	6	20
ed_smith@ca.ibm.com	6	5
duongn@ca.ibm.com	6	4
Boris_Bokowski@ca.ibm.com	6	2
obesedin@ca.ibm.com	6	2
jfogell@us.ibm.com	6	1
webmaster@eclipse.org	6	1
frankb@ca.ibm.com	6	1
ERCP-inbox@eclipse.org	6	1
Karice_McIntyre@ca.ibm.com	6	1
pwebster@ca.ibm.com	6	1
ggayed@ca.ibm.com	6	1
CLASS 7		
Michael_Valenta@ca.ibm.com	7	512
Michael_Valenta@oti.com	7	304
Tomasz.Zarna@pl.ibm.com	7	83
Szymon.Brandys@pl.ibm.com	7	11
krzysztof.daniel@gmail.com	7	3
krzysztof.michalski@pl.ibm.com	7	2

Pawel.Pogorzelski@pl.ibm.com	7	2
Larry.Isaacs@sas.com	7	2

Appendix B: Priority of developers

Class	Email	Priority (hours in last Resolution)
1	jeff-bugs@code9.com	5997.2
1	John_Arthorne@oti.com	20925.33333
1	Knut_Radloff@oti.com	22264.08334
1	john_arthorne@ca.ibm.com	27661.9
1	knut_radloff@us.ibm.com	29726.16667
1	steve_northover@ca.ibm.com	36686.25001
1	James_Moody@ca.ibm.com	57161.6
1	Lynne_Kues@oti.com	70024.40001
1	James_Moody@oti.com	84774.8
1	Steve_Northover@oti.com	123182.4833
2	Debbie_Wilson@oti.com	14.649998
2	debbie_wilson@ca.ibm.com	3207.566664
2	Kevin_McGuire@ca.ibm.com	5623.166662
2	Jean-Michel_Lemieux@oti.com	8124.899995

2	dj_houghton@ca.ibm.com	16729.96667
2	Mike_Wilson@ca.ibm.com	22295.08334
2	jean-michel_lemieux@ca.ibm.com	50117.58333
2	DJ_Houghton@oti.com	94797.65
2	Kevin_McGuire@oti.com	182512.8333
2	Mike_Wilson@oti.com	366574.4833
3	rodrigo_peretti@ca.ibm.com	0.016667
3	Kevin_Haaland@ca.ibm.com	70.9
3	lynne_kues2@us.ibm.com	119.466667
3	erich_gamma@ch.ibm.com	184.933334
3	Rodrigo_Peretti@oti.com	360.416667
3	Kai-Uwe_Maetzel@oti.com	1142.833333
3	Eduardo_Pereira@oti.com	1841.966666
3	Rodrigo_Peretti2@oti.com	8143.383334
3	jeff_mcaffer@ca.ibm.com	8577.383333
3	Jeff_McAffer@oti.com	8660.749997
3	lynne_kues@us.ibm.com	50690.83333
3	Kevin_Haaland@oti.com	58733.70001
3	Carolyn_MacLeod@oti.com	66300.08333
3	Andre_Weinand@oti.com	75728.51667

3	Carolyn_MacLeod@ca.ibm.com	92200.55
3	Christophe_Cornu@oti.com	103894.7667
3	andre_weinand@ch.ibm.com	114611.1833
3	Boris_Shingarov@oti.com	212745.7167
3	christophe_cornu@ca.ibm.com	223660.8333
4	mike_mallett@ca.ibm.com	0.016667
4	nick_edgar@ca.ibm.com	1
4	chris_mclaren@ca.ibm.com	5.466667
4	Charlie_Gracie@oti.com	22.2
4	Shantha_Ramachandran@oti.com	27.266667
4	birsan@ca.ibm.com	66.55
4	Jim_des_Rivieres@oti.com	86.933333
4	rchaves.eclipse@gmail.com	108.6
4	Sebastian_Peleato@ca.ibm.com	115.299999
4	kcornell@ca.ibm.com	174.883333
4	max_sigalov@ca.ibm.com	193.466667
4	Olivier_Thomann@oti.com	217.15
4	cdm@qnx.com	484.950001
4	johnrose@austin.ibm.com	520.63333
4	lkues@us.ibm.com	651.25

4	Daniel_Megert@oti.com	700.983333
4	lindaw@ca.ibm.com	770.16666
4	douglas.pollock@magma.ca	826.5
4	Chris_McLaren@oti.com	894.6666
4	jdt-ui-inbox@eclipse.org	966.75
4	Tim_Ellison@uk.ibm.com	1158.666667
4	Jed_Anderson@us.ibm.com	1208.883333
4	jdt-core-inbox@eclipse.org	1378.566666
4	Darin_Swanson@oti.com	1532.766667
4	Rafael_Chaves@oti.com	2108.999999
4	Dirk_Baeumer@oti.com	2249.333333
4	sridhar@ics.com	2256.533334
4	Tod_Creasey@oti.com	2664.933333
4	silvio_boehler@ca.ibm.com	2972.166666
4	simon_arsenault@ca.ibm.com	3057.9
4	pascal_rapicault@ca.ibm.com	3078.666665
4	jeff_brown@oti.com	3263.833334
4	bshingar@ca.ibm.com	3377.783332
4	pde-ui-inbox@eclipse.org	4214.333333
4	mvm@ca.ibm.com	4335.416667
4	tjwatson@us.ibm.com	4884.733337

4	rafael_chaves@ca.ibm.com	5146.016665
4	kcornell@rational.com	5384.816666
4	wassimm@ca.ibm.com	6026
4	Philippe_Mulet@oti.com	6358.183333
4	Tod_Creasey@ca.ibm.com	8277.966666
4	Nick_Edgar@oti.com	8918.699998
4	pde-doc-inbox@eclipse.org	9558.05
4	dean_roberts@ca.ibm.com	9706.266666
4	klicnik@ca.ibm.com	10219.23334
4	eclipse.rc@gmail.com	10251.16667
4	jed.anderson@genuitec.com	11265.53333
4	dejan@ca.ibm.com	11995.4
4	equinox-inbox@eclipse.org	15111.55
4	gheorghe@ca.ibm.com	18593.93333
4	zhiyongl@ca.ibm.com	25309.63334
4	Jed_Anderson@oti.com	27395.06667
4	Erich_Gamma@oti.com	29264.86667
4	Simon_Arsenault@oti.com	30094.45
4	Antonio_D'souza@oti.com	31235.3
4	Adam_Kiezun@oti.com	31980.48333
4	curtis_d'entremont@oti.com	40558.15

4	ymnk@jcraft.com	47652.23334
4	Felipe_Heidrich@oti.com	59660.73333
4	felipe_heidrich@ca.ibm.com	230751.75
5	Platform-UI-Inbox@eclipse.org	2318.583334
5	platform-doc-inbox@eclipse.org	11046.08333
5	platform-text-inbox@eclipse.or	16720.81667
5	platform-resources-inbox@eclip	27278.86667
5	platform-runtime-inbox@eclipse	35271.21666
5	platform-core-inbox@eclipse.or	69353.64999
5	platform-cvs-inbox@eclipse.org	82547.33334
5	platform-swt-inbox@eclipse.org	108713.65
5	platform-team-inbox@eclipse.or	154691
5	platform-compare-inbox@eclipse	159562.0333
5	platform-vcm-inbox@eclipse.org	383722.2833
6	jfogell@us.ibm.com	23.783334
6	obesedin@ca.ibm.com	43.383333
6	ggayed@ca.ibm.com	140.966666
6	duongn@ca.ibm.com	553.53334
6	webmaster@eclipse.org	601.783333

6	pwebster@ca.ibm.com	1517.633333
6	Karice_McIntyre@ca.ibm.com	2055.05
6	Boris_Bokowski@ca.ibm.com	2756.033333
6	ed_smith@ca.ibm.com	2902.583333
6	ERCP-inbox@eclipse.org	9098.116667
6	frankb@ca.ibm.com	11225.31667
6	krbarnes@ca.ibm.com	11872.8
6	bbiggs@ca.ibm.com	13125.46668
6	billy.biggs@eclipse.org	20971.76667
6	Grant_Gayed@oti.com	22657.73334
6	grant_gayed@ca.ibm.com	28303.51666
6	Silenio_Quarti@oti.com	45794.08331
6	Veronika_Irvine@oti.com	65173.5
6	Silenio_Quarti@ca.ibm.com	84371.43333
6	veronika_irvine@ca.ibm.com	242322.55
7	Pawel.Pogorzelski@pl.ibm.com	4.9
7	Larry.Isaacs@sas.com	186.316667
7	krzysztof.daniel@gmail.com	1005.05
7	krzysztof.michalski@pl.ibm.com	1783.166667
7	Szymon.Brandys@pl.ibm.com	2327.916666

7	Tomasz.Zarna@pl.ibm.com	27768.98334
7	Michael_Valenta@oti.com	87696.48334
7	Michael_Valenta@ca.ibm.com	106909.95

Appendix C: Matlab code

```
clearvars;

load('eclipse_bugs.mat');

tic

data = eclipseplatform bugs11;

n = length(data);

[rows cols] = size(data);

X = data(:,[2:49]);

Y = data(:,50);

% Use cvpartition to separate the dataset into a test set and a training set

% cvpartition will automatically ensure that feature values are evenly

% divided across the test set and the training set.

% Create a cvpartition object that defined the folds

c = cvpartition(Y,'Kfold',10);

% Create a training set

X_Train = X(training(c,1),:);

Y_Train = Y(training(c,1));

% Train a Classifier using the Training Set

var2 = var(data(:,1:49));

RF = fitcnb(X_Train,Y_Train);

[RF_Predicted] = RF.predict(X(test(c,1),:));

% yPredicted = str2double(RF_Predicted);

[conf, classorder] = confusionmat(Y(test(c,1)),RF_Predicted);

% False_Alarm _Rate = false detections/all detections =FP/(TN+FP)
```

diagonal = diag(conf);

False_alarm _rate = 1-sum (diagonal)/n

no_correctly_classified_sam ples = sum (diagonal)

accuracy_percenta ge = (no_correctly_classified_sam ples/n)*100

Appendix D: Weka software results for clustering

D1: naive bayes

```
Correctly Classified Instances        1144               49.0776 %
Incorrectly Classified Instances      1187               50.9224 %
Kappa statistic                          0.4012
Mean absolute error                      0.1521
Root mean squared error                  0.3414
Relative absolute error                 62.792  %
Root relative squared error             98.1806 %
Total Number of Instances             2331

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.285    0.063    0.452      0.285    0.350      0.270    0.771     0.384     1
                 0.580    0.080    0.571      0.580    0.575      0.497    0.858     0.621     2
                 0.401    0.058    0.429      0.401    0.415      0.354    0.835     0.391     3
                 0.224    0.086    0.232      0.224    0.228      0.141    0.705     0.200     4
                 0.562    0.100    0.609      0.562    0.585      0.477    0.864     0.665     5
                 0.606    0.077    0.607      0.606    0.607      0.529    0.918     0.630     6
                 0.675    0.131    0.385      0.675    0.490      0.431    0.876     0.427     7
Weighted Avg.    0.491    0.085    0.498      0.491    0.487      0.405    0.839     0.508

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 103  65  29  44  42  40  38 |   a = 1
  31 210  11  12  29   1  68 |   b = 2
  17  20  91  28  14  40  17 |   c = 3
  17   9  31  54  46  41  43 |   d = 4
  24  20   9  43 284  28  97 |   e = 5
  27   8  38  39  30 232   9 |   f = 6
   9  36   3  13  21   0 170 |   g = 7
```

D2: simpleLogistic

```
Correctly Classified Instances        4761               61.2741 %
Incorrectly Classified Instances      3009               38.7259 %
Kappa statistic                          0.5411
Mean absolute error                      0.1457
Root mean squared error                  0.2704
Relative absolute error                 60.1158 %
Root relative squared error             77.6939 %
Total Number of Instances             7770

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.498    0.070    0.566      0.498    0.530      0.451    0.866     0.570     1
                 0.641    0.061    0.651      0.641    0.646      0.583    0.925     0.742     2
                 0.423    0.033    0.572      0.423    0.486      0.447    0.903     0.573     3
                 0.270    0.038    0.462      0.270    0.341      0.296    0.803     0.355     4
                 0.761    0.079    0.715      0.761    0.737      0.667    0.941     0.834     5
                 0.813    0.110    0.594      0.813    0.687      0.625    0.939     0.716     6
                 0.655    0.065    0.573      0.655    0.611      0.557    0.941     0.653     7
Weighted Avg.    0.613    0.070    0.605      0.613    0.601      0.540    0.908     0.662

=== Confusion Matrix ===

    a    b    c    d    e    f    g   <-- classified as
  600  170   55   51   55  205   70 |    a = 1
  119  752   13   23   71   39  156 |    b = 2
   85   21  311   59   34  210   15 |    c = 3
   84   75   66  228  151  178   63 |    d = 4
   52   27   23   69 1226   75  139 |    e = 5
   89    9   72   41   23 1042    5 |    f = 6
   31  102    4   22  154    4  602 |    g = 7
```

D3: random forest

```
Correctly Classified Instances        3993               51.39   %
Incorrectly Classified Instances      3777               48.61   %
Kappa statistic                          0.4263
Mean absolute error                      0.1407
Root mean squared error                  0.3551
Relative absolute error                 58.0775 %
Root relative squared error            102.0085 %
Total Number of Instances             7770

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.490    0.110    0.450      0.490    0.469      0.367    0.713     0.343     1
                 0.592    0.075    0.584      0.592    0.588      0.514    0.781     0.460     2
                 0.418    0.057    0.435      0.418    0.426      0.368    0.711     0.276     3
                 0.327    0.072    0.358      0.327    0.341      0.265    0.648     0.216     4
                 0.616    0.102    0.613      0.616    0.614      0.513    0.778     0.504     5
                 0.532    0.093    0.530      0.532    0.531      0.438    0.751     0.406     6
                 0.491    0.064    0.508      0.491    0.499      0.434    0.744     0.355     7
Weighted Avg.    0.514    0.085    0.513      0.514    0.513      0.428    0.739     0.386

=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 591 136  68  89 105 146  71 |   a = 1
 169 695  28  49  64  42 126 |   b = 2
  82  44 307  68  59 157  18 |   c = 3
  92  60  70 276 138 144  65 |   d = 4
 127  89  43 139 992  87 134 |   e = 5
 171  42 170 102  93 681  22 |   f = 6
  81 124  19  49 167  28 451 |   g = 7
```

D4: J48

```
Correctly Classified Instances        4758               61.2355 %
Incorrectly Classified Instances      3012               38.7645 %
Kappa statistic                          0.5421
Mean absolute error                      0.1241
Root mean squared error                  0.2954
Relative absolute error                 51.1985 %
Root relative squared error             84.8585 %
Total Number of Instances             7770

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.539    0.088    0.529      0.539    0.534      0.447    0.796     0.468     1
                 0.691    0.059    0.674      0.691    0.682      0.625    0.878     0.666     2
                 0.503    0.039    0.573      0.503    0.536      0.492    0.840     0.471     3
                 0.401    0.057    0.462      0.401    0.430      0.367    0.759     0.341     4
                 0.713    0.074    0.715      0.713    0.714      0.639    0.876     0.662     5
                 0.687    0.087    0.609      0.687    0.646      0.572    0.861     0.565     6
                 0.609    0.051    0.616      0.609    0.613      0.561    0.873     0.545     7
Weighted Avg.    0.612    0.068    0.610      0.612    0.610      0.543    0.845     0.550

=== Confusion Matrix ===

    a    b    c    d    e    f    g   <-- classified as
  650  123   56   76   74  170   57 |    a = 1
  115  810    8   32   66   26  116 |    b = 2
   65   17  370   70   27  181    5 |    c = 3
  108   57   59  339  115  130   37 |    d = 4
   85   71   28   90 1149   55  133 |    e = 5
  135   25  119   88   33  880    1 |    f = 6
   71   99    6   38  143    2  560 |    g = 7
```

D5: ANN

```
Correctly Classified Instances        5097               65.5985 %
Incorrectly Classified Instances      2673               34.4015 %
Kappa statistic                          0.592
Mean absolute error                      0.1497
Root mean squared error                  0.2621
Relative absolute error                 61.7915 %
Root relative squared error             75.3002 %
Total Number of Instances             7770

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.570    0.063    0.625      0.570    0.596      0.527    0.893     0.675     1
                 0.698    0.056    0.691      0.698    0.694      0.640    0.936     0.790     2
                 0.513    0.028    0.656      0.513    0.576      0.542    0.913     0.641     3
                 0.375    0.029    0.611      0.375    0.465      0.431    0.856     0.515     4
                 0.809    0.084    0.717      0.809    0.760      0.694    0.945     0.844     5
                 0.793    0.097    0.617      0.793    0.694      0.632    0.943     0.751     6
                 0.629    0.051    0.625      0.629    0.627      0.577    0.934     0.663     7
Weighted Avg.    0.656    0.063    0.654      0.656    0.648      0.593    0.921     0.718

=== Confusion Matrix ===

    a    b    c    d    e    f    g   <-- classified as
  687  124   46   40   77  177   55 |    a = 1
   93  819   14   24   83   23  117 |    b = 2
   50   28  377   34   31  197   18 |    c = 3
   73   52   48  317  139  171   45 |    d = 4
   53   38    9   41 1303   59  108 |    e = 5
   90   21   75   40   35 1016    4 |    f = 6
   53  104    6   23  150    5  578 |    g = 7
```

CURRICULUM VITAE

Name: Ali Jaafar Meera ALARKAWAZI

Study:

- 2016: B.Sc. in Computer Science, Al-Rafidain University College, Baghdad-Iraq.
- 2021: Finished my M.Sc. in Computer Engineering, Altinbaş University, Istanbul-Turkey.