Analyzing the Performance of Hybrid Model Reconstruction Approach for balanced and unbalanced Dataset

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

Master of Technology
in
Computer Science and Engineering

Under the esteemed guidance of
Dr. Ruchika Malhotra
(Associate Head and Assistant Professor – Computer Science and Engineering)
Delhi Technological University

Submitted By-
Aakansha Yadav
(Roll No. - 2K15/CSE/01)

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

SESSION: 2015-2017

CERTIFICATE

This is to certify that report entitled Aakansha Yadav (2K15/CSE/01) has completed the thesis titled "**Analyzing the Performance of Hybrid Model Reconstruction Approach for balanced and unbalanced Dataset**" under my supervision in partial fulfilment of the MASTER OF TECHNOLOGY degree in Computer Science Engineering at DELHI TECHNOLOGICAL UNIVERSITY.

Supervisor

Dr.Ruchika Malhotra
Associate Head and Assistant Professor
Department of Computer Science and Engineering
Delhi Technological University
Delhi -110042

DECLARATION

We hereby declare that the thesis work entitled "**Analyzing the Performance of Hybrid Model Reconstruction Approach for balanced and unbalanced Dataset**" which is being submitted to Delhi Technological University, in partial fulfilment of requirements for the award of degree of Master of Technology (Computer Science Engineering) is a bonafidereport of thesis carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any degree.

Aakansha Yadav
2K15/CSE/01

ACKNOWLEDGEMENT

I am very thankful to Dr.Ruchika Malhotra (Assistant Professor, Computer Science Eng. Dept.) and all the faculty members of the Computer Science Engineering Dept. of DTU. They all provided immense support and guidance for the completion of the project undertaken by me.

I would also like to express my gratitude to the university for providing the laboratories, infrastructure, testing facilities and environment which allowed me to work without any obstructions.

I would also like to appreciate the support provided by our lab assistants, seniors and peer group who aided me with all the knowledge they had regarding various topics.

Aakansha Yadav

M. Tech. in Computer Science Engineering

Roll No. 2K15/CSE/01

ABSTRACT

Software defect prediction helps in identifying fault prone classes of a software in the early phases of software development life cycle. This helps in efficient resource allocation, since more resources should be allocated to such fault prone classes. Defect prediction(DP) models are developed from different machine learning methodologies in which the models are trained using data from previous releases of a software. However, while developing DP models, researchers have to deal with certain issues. Two such critical issues are addressed in this study a) unavailability of historical data of a software project, and b) imbalanced nature of training data set, where the distribution of classes is highly skewed. In order to deal with scarcity of historical data of a project, literature studies use cross project defect prediction(CPDP). Though, a number of literature studies have suggested methods for improving the accuracy of DP models with imbalanced datasets, but the imbalanced issue has not been properly addressed in the scenario of CPDP.Thus, this study analyzes the performance of a CPDP model HYDRA(Hybridized moDel Reconstruction Approach), developed by Xia et al. using imbalanced data. The results of the study are empirically validated using twenty open source software projects, where ten software projects are of imbalanced nature. Furthermore, we also suggest variation in the fitness function of HYDRA for improving its performance. The results of the study are statistically assessed using Wilcoxon test.

TABLE OF CONTENTS

FIGURES AND TABLES

CHAPTER 1

Introduction

Software quality today has become one of the most crucial aspects of software development. A defective software product leads to unsatisfied customers. Thus, it is essential to develop efficient DP models to anticipate flaws in the initial stages of software development life cycle.Over the years, DP has proven to be a useful aid in organizing a project's testing facilities and resources [6, 63, 64]. For instance, effective DP models can guide the testing personnel to cater to classes which are more likely to be defective than the others. Recently, a large number of studies pertaining to DP have been conducted so as to provide accurate and effective DP models [2, 9, 10, 12, 20, 45]. However, most of the past studies have focused on constructing DP models by employing the training set designed from the historical data of the project itself.The defects have been predicted in the later releases of the same project, or have provided the results of k-fold cross-validation on that data set itself [34, 48]. A major shortcoming of this approach is that in order to construct such a DP model, our primary requirement is an appropriate amount of historical data of the project under consideration. However,in reality, historical data that fully satisfies our requirements is not always available, because either such kind of historical data has not been populated at all, or was not collected effectively [33, 59]. This implies that it is not always feasible to conduct DP studies based on the historical data of a project. The scarcity of within project historical data can be overcome by using CPDP.

CPDP is asignificant research area in software engineering, wherein we thrive to predict faultsin a given project using DP models constructed by employing appropriate training data obtained from history of other projects [54, 68].In case of unavailability of historical data of a project, a wide range of public data sets are available, which may be potentially employed as training datasets for developing DP models.

Apart from having lack of historical data, another critical issue faced by researchers while developing DP models is the imbalanced nature of training data. In an imbalanced dataset, instances of one type of class are present in large numbers as compared to that of other type. In such a scenario, the model will not be able to train properly which may lead to inaccurate results. Though, past studies have proposed various measures for proper model learning using an imbalanced within project data, the same has not been explored for CPDP. Therefore, there is an urgent need to analyze the performance of the established CPDP models using imbalanced training data.

HYDRA is a model for CPDP [65].It uses training data which has previously labeled classes as clean or buggy from multiple projects to predict the clean/buggy nature of unlabeled classes in the target project. The model aggregates the result of multiple classifiers which are

trained on different projects to predict buggy instances in the target project. This study empirically evaluates 20 widely used open source datasets out of which 10 are imbalanced, using the HYDRA model. We are dealing with highly to moderately imbalanced datasets where the buggy classes are less in comparison with non-buggy classes. Since the number of buggy classes is few,the model will not be able to learn much about these classes. As a result,the predictions made by the model may be inaccurate.The results provided by HYDRA on both types ofdatasets(balanced and imbalanced) are assessed by using two stable performance metrics which are F1 score and AUC(Area Under the receiver operating characteristic Curve) measure.

Furthermore, in the past there have been studies that have proved that there is variation in the performance of the prediction models with change in fitness function [1, 5, 16]. Therefore, with an intend of improving the performance of HYDRA, the study evaluates its performance using a different fitness function, i.e. using AUC measure for fitness evaluation. The previously proposed model of HYDRA used F1 score as the fitness function.

In order to conduct the various experiments, we are using 10 datasets each of balanced and imbalanced nature from PROMISE repository. The results are statistically analyzed using Wilcoxon test. The results establish the effectiveness of HYDRA with the use of imbalanced dataset for CPDP. Furthermore, the study supports the use of AUC measure as the fitness function for an improved performance of HYDRA.

1.1 *Research Questions*

In this study the following research questions are explored:

- ***RQ1****: What is the performance of HYDRA using balanced and imbalanced datasets?*

  The performance of HYDRA on both balanced and imbalanced datasets is evaluated. Ten balanced and ten unbalanced data sets each are used for doing so. The obtained results are evaluated using F1 score and AUC measure.

- ***RQ2****: Does the results of HYDRA vary with change in fitness function?*

  HYDRA uses GA in its process to build the classifier. Fitness function should be carefully chosen for better performance of an evolutionary algorithm such as GA. We are using F1 score and AUC measure as the fitness functions and analyze the change in the performance of the model. The aim is to suggest improvement in the performance of HYDRA model with variation in fitness function.

1.2*Motivation of study*

Defect prediction has proved to be very useful as it helps in predicting software components that are defect prone. Thus, it helps in reducing the resources that are required and ultimately the cost. Defect prediction has been further improved to cross project defect prediction in cases where a software has less historical data. Most of the prediction models that are build uses balanced dataset but in most practical cases the data is generally imbalanced. So, we

need to find out that whether these models perform equally well using imbalanced dataset also. So, our work focuses on imbalanced dataset and comparing the accuracy of prediction model with balanced dataset. For this purpose we are using the model build by Xin Xia. We have implemented the model hydra and then experiment is done using both balanced and imbalanced dataset.

Further, we are also trying to improve the performance of HYDRA. Hydra uses genetic algorithm. We are experimenting using different performance metrics as the fitness function and analyzing the performance of hydra. The aim is to improve the accuracy of the model and also its accuracy on imbalanced dataset. We are also analyzing the time taken by the model using the different datasets. A balanced dataset has equal number of instances for both of its classes in case of binary classification. In our case the classes are clean and buggy. It is a binary classification. So, if the instances of a software has uniform distribution for both the classes than it is a balanced dataset. Whereas in case of imbalanced dataset, more instances belong to one class itself. In most of the practical cases, the dataset is imbalanced. In case of binary classification having two classes as clean and buggy. In practical cases, there are few instances that are buggy and most of the instances are clean. This leads to imbalanced dataset as most of the instances belong to one class only. So, it is necessary that a prediction model gives good accuracy in case of imbalanced dataset also. The accuracy of hydra has been formulated using balanced dataset[1]. In this research we are analyzing the performance of hydra using imbalanced dataset and comparing its accuracy with that of balanced dataset. We will observe that if a model for cross project defect prediction has good accuracy on balanced dataset then how will it perform on imbalanced dataset. Performance measures should be selected carefully while working with imbalanced dataset. Accuracy is not a correct measure for imbalanced dataset as it gives biased result and misguides the performance of the model. F1-score is considered to be one of the performance measure that provide correct results on imbalanced dataset as well. So, we are using F1-score for comparing the performance of the model using both the datasets(balanced and imbalanced).

Fitness function in genetic algorithm is used as a measure to select chromosome from the initial population. It is used such that it chooses the best solution from the initial population. In our model we need to select the solution that provides good accuracy of the model , so such a factor should be used as fitness function so that it chooses the solution that provides the best accuracy. In the model that we are using uses F1-score as the fitness function. In this research question we are analyzing whether the use of different performance measures as fitness function affects the performance of hydra. We are comparing the performance using

two performance measures. One is the F1-score and other is Area under the ROC-Curve(AUC-Curve). AUC curve is defined as the plot of sensitivity versus specificity, by taking sensitivity on the y-axis and specificity on the x-axis. It is an effective method for analyzing the quality or performance of the described prediction model.

1.4 Organization of thesis

This thesis is organized in various sections as follows: Section 2 gives summary of the related literature, Section 3 summarizes the research methodologies used in this paper including overview of the framework and the description of HYDRA. Section 4 describes empirical study design i.e. dataset description, variable selection, validation method and performance evaluation metrics. Section 5 states the results of the study which has been discussed with correspondence to each RQ. Section 6 states the various validity threats and at last the section 7 summarizes the research work's conclusion and suggests some future work

CHAPTER 2

Literature Review

This section gives an overview of the research work done with relation to our study. It includes the past studies related to CPDP models and the DP models developed using imbalanced datasets

2.1 Cross Project Defect Prediction Models

Various DP models have been designed and proposed in the past. There have been studies which establish the relationship between Object Oriented (OO) metrics and fault proneness [2, 3, 9, 13, 20, 29, 31, 57]. Moreover, the importance of machine learning to develop DP models has been increasing ever since [18, 40, 41, 42]. Various researchers have also performed a comparison of multiple machine learning techniques on the basis of their ability to design effective models for software DP[14, 40].

These DP models are trained using historical data of a project. However, sometimes less historical data is available due to which the models are not trained properly. Due to lack of historical data, CPDP models have been found advantageous. A lot of work has been done by researchers in this field [24, 25, 30, 49, 50, 52, 53]. Work on the feasibility of CPDP models and the effect of mixed project data on learning algorithmhas been done[22]. Models based on transfer learning have also been built [38, 50]. Herbold [24] and Hosseini et al. [25] worked on different strategies for selecting the training data for CPDP models. Ryu et al.[54] showed the efficiency of support vector machine while developing CPDP models. A study by Ryu and Baik[52] used an approach of multi-objective naive bayes algorithm for training the CPDP model.By the use of multi objective the probability of detection of buggy instances was maximized. Also, semi-supervised and unsupervised learning for CPDP models has been proposed [67].

Zimmerman et al.[68] carried out a large scale experiment on finding out the factor that affects the most while selecting any project as training project for CPDP. It was an experiment comparing the importance of data, domain and process while selecting the training project. Turhanet al.[59]studied data for training from the same company itself as well as from other companiesfor DP model. In his study, data from within-company gave better results as compared to cross-company data. Jureczko and Madeyski[28] formed

clusters of the projects having similar features. They confirmed that in case of absence of historical data, predictors from within the same cluster can be used.

Cruz and Ochimizu [6] used software metrics as features for building a model with logistic regression as the underlying machine learning classifier and worked on the correctness of the model using various projects. Turhan at el.[58, 59] studied another related issue of cross-company DP i.e. predicting defects of local projects by using the projects of other companies as training data. It was concluded that data from cross-company projects enhances the defect detection probability, but it also increases the false positive rate.In such a scenario, they recommended filtering of cross company data through nearest neighbor. Watanabe et al. [62] introduced a method known as "metrics compensation" so as to try to adapt the predictor built from a java based project and apply it to a C++ based project. It was claimed that it may be possible that we can reuse a predictor among projects created using varied programming languages but only foranalogous domain and analogous size. It stated that the characteristics of data are possibly a major factor in CPDP models. However, this conclusion has weak generalization ability because the experiment was carried out using only two applications. Many more experiments on generalization ability of CPDP models have been carried out [25, 30].

2.2 Defect Prediction Models using Imbalanced Dataset

Studies have been conducted whichanalyze the stability of DP models by using datasets with class imbalanceissue [32, 35, 36, 37, 66]. The experiment was done with six different prediction models and the performance of each model on imbalanced dataset was investigated[39]. It showed that the models developed using Random forest and Naïve Bayes were more stable as compared to other models and C4.5 showed to be the most unstable model on imbalanced dataset. Another study using imbalanced data for the stability of feature selection was conducted [56]. Different software metrics as features were used in most of the models. They worked on 18 such software metrics and analyzed their stability on imbalanced dataset. Much workis done on resampling, instance weighting and ensemble methods for dealing with imbalanced dataset in DP models [51, 55]. Also,a study on hybrid sampling strategy to improve the learning in case of imbalanced training data was conducted [47]. Other studies on software DP using imbalanced dataset were also conducted [4, 15, 17, 21, 26, 36, 60, 61]. Various robust performance measures have also been studied, which provide stability for classification using imbalanced dataset [19, 21, 36].

However, there have been very few studies on CPDP models with imbalanced datasets[27]. So, in this study we are investigating the performance of a CPDP model on imbalanced dataset. Our work is inspired by the work of Xia et al. [65]. They developed a model for CPDP using Genetic Algorithm (GA) and logistic regression as the machine learning classifier.The model has better accuracy results as compared to other recently proposed approaches [65]. They usedbalanced datasets from promise repository for DP. However, in this paper we are also evaluating the performance of HYDRA on imbalanced datasets. Practically, in most of the software datasets, there are fewer buggy classes than clean classes. So, this leads to imbalanced nature of the dataset which misguides the accuracy of any model. Thus, this study performs experiments using imbalanced datasets and compares the results with that of balanced datasets. Furthermore, we are trying to improve the existing model of HYDRA by changing the fitness function of GA.

CHAPTER 3

Research Methodology

This chapter presents the experimental framework used in this work for CPDP.

Fig.3.1 shows the framework of the model HYDRA as Proposed by Xia et al. [66].

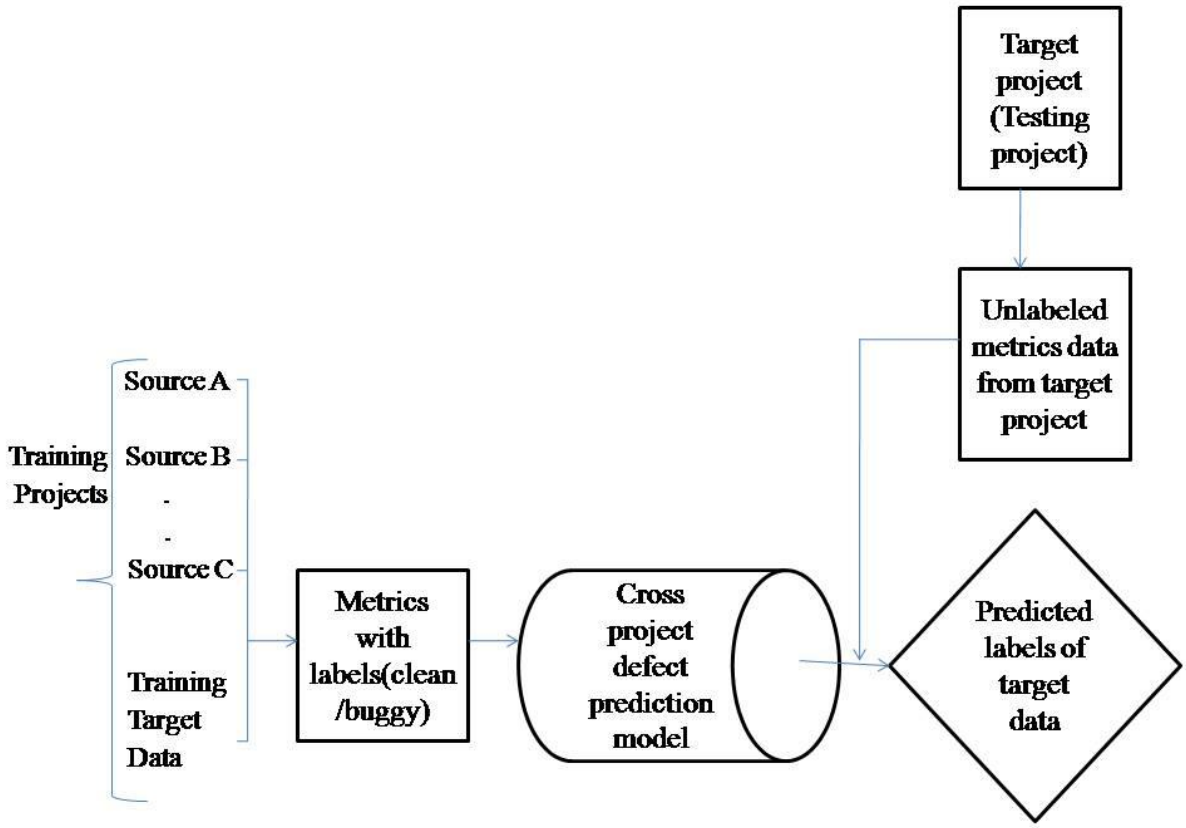

Fig. 3.1 Framework of model HYDRA

Training datasets are the source projects with labeledinstances(clean or buggy) that are used to train a model. Testing dataset is the project of whom we have to predict the label of the instances.Suppose there are 'N' projects, known as source projects $\{S_1, S_2 \ldots S_N\}$ and a target project 'T'. We consider 'N' source projects along with 5% labeled instances from the target project(known as training target data, T_t) as the training dataset of our model, where an

instance is an OO class of the project. The unlabeled instances of the target project 'T' is the testing dataset.OO metrics of instances of a project are used as the features to train our CPDP model. So, in our training dataset each instance is represented by 20 OO metrics (refer Section 4.1) along with their label (1 if buggy and 0 if clean). Each instance of testing dataset is also represented using these 20 OO metrics which is provided as input to our model and the model predicts the label of the instance as1 or 0.

Further, the model involves many steps. For each of the source project and the training target data(T_t), a machine learning classification model(M) is developed which uses logistic regression as the underlying classifier. Hence, we have (N+1) classification models. Each model outputs a likelihood score which shows the probability of an instance of being defective. These models are assigned weights according to the performance of each model i.e. the model with high F1 score will have higher weight. Initially random weights are assigned to the classifiers. These weights are optimized with the help of Genetic Algorithm(GA).

- ***Genetic Algorithm***

Fig. 3.2 shows the steps involved in a genetic algorithm.

Genetic Algorithm

1. pop_size ← number of chromosomes
2. max_gen ← maximum number of generations
3. P ← initial population with pop-size members
4. Evaluate P and record the best solution(having maximum F1 score);
5. Let cur_gen =0 and P' = P
6. **While**cur_gen ≤ max_gen
    a. Selection: Roulette wheel selection procedure is used for selection(Goldberg and Holland 1998). It select parents with highest fitness score which is F1 score on T_t in our study.
    b. Crossover: Single point crossover operator is used.
    c. Mutation: Random mutation is used. It randomly swaps the value of gene with a certain probability to another value in its range.
7. Evaluate P' and record best solution.
8. Cur_gen = cur_gen + 1
9. **End while**
10. Output: $\sum_{i=1}^{N+1} \alpha_i M_i$,threshold which achieves the highest F1 score.

Fig. 3.2 Genetic Algorithm

The initial population(P) of GA is made up of pop_size chromosomes(i.e. solutions). Each chromosome consists of (N+2) genes, where genes are the weights assigned to the classifiers. The first (N+1) values lies between 0 to 1 whereas the $(N+2)^{th}$ value represents the threshold which is a user-defined level that is used to decide whether an instance is buggy or clean and its value lies between 1 to (N+1). 'P' is evaluated and the best solution is recorded i.e. having the highest F1 score(fitness function). After that selection,crossover and mutation operations are applied on the initial population max_gen times. Every time we evaluate 'P' and record the best solution. So, the final output of GA gives a solution with the maximum F1 score i.e. we have the weights of all the classifiers along with the threshold that provides us with highest F1 score. This combination of all the classifiers along with their weights is known as GA classifier.

- ***GA Classifier***

GA classifier provides us with the final label of the instances of target project.

Label(i) = { 1 (i.e. buggy) if comp(j) ≥ threshold

0(i.e. clean) otherwise }

Where,

$$\text{Comp(j)} = \frac{\sum_{i=1}^{N+1} \alpha_i \times Score_i(j)}{L}$$

In the above equation,

α_i(i=1 to N+1) are the weights of the classifiers.

$Score_i(j)$ is the likelihood score given by i^{th}classifier for instance j.

LOC is the no. of lines of code.

.

CHAPTER 4

Empirical Study Design

This chapter discuss different design consideration of our thesis.

1.1 Variable Selection

This section describes the dependent and independent variables of the study.

In our study, fault proneness is the dependent variable.It is the probability of detecting a bug in an instance i.e. an OO class [7]. We have taken it as a binary variable because our focus is to predict whether a class is faulty or not. A software component is known as faulty if it contains some error or bug which can lead to incorrect or unexpected results. On the contrary, a component free of any such error or bug is known to be clean. We have taken various OO metrics as the independent variables. The incorporated metrics are listed below:

S. R. Chidamber and C. F. Kemerer [8]:

- WMC(Weighted Methods for Class): Represents the total number of methods of an instance.
- DIT(Depth of Inheritance tree): It measures the longest distance from a given class to the root of an inheritance tree.
- NOC(Number of Children): Given an inheritance tree of a class it returns the totalnumber of children.
- CBO(Coupling between Objects): Enumerates the instances coupled to a particular instance.
- RFC(Response for Class): It is the number of distinct methods invoked by code in a given class.
- LCOM(Lack of Cohesion of Methods): The number of method pairs in a class that do not share access to any class attributes.

J. Bansiya and C. G. Davis metrics [2]:

- NPM(Number of Public Methods): Determines the public methods in a given instance.
- LOC(Lines of Code): Counts the total lines of code in a given class.

- DAM(Data Access Metric): Measures the fraction of private/protected attributes and the total attributes in a given instance.
- MOA(Measure of Aggregation): Determines the total attributes in a given class which are of user-defined types.
- MFA(Measure of Functional Abstraction): Defines the total methods inherited by a given class divided by the methods that can be accessed by the member methods of the given class.
- CAM(Cohesion among Methods of Class): Defines the ratio of the sum of the number of different parameter types of every method to the product of total methods and the different method parameter types in the whole class.

M. Tang, M. Kao, and M. Chen metrices [58]:

- IC(Inheritance Coupling): Given a class, it returns total parent classes that it is coupled to.
- CBM(Coupling between Methods): Defines the new or overwritten methods that all inherited methods in a given class are coupled to.
- AMC(Average Method Complexity): Defines the average size of methods in a given instance.

R. Martin metrics [44]:

- Ca(Afferent Coupling): Given a class, it returns the total classes that are dependent on it.
- Ce(Efferent Coupling): It returns the total classes that a given class depends upon.

T. McCabe metrics(1976):

- max_cc(maximum cyclomatic complexity): counts the maximum McCabe's cyclomatic complexity score of methods in a given class.
- avg_cc(average cyclomatic complexity): counts the arithmetic mean of the McCabe's cyclomatic complexity scores of methods in a given class.

B. Henderson-Sellers metrics [23]:

- LCOM3(Lack of Cohesion of Methods Version 3): Another type of LCOM metric proposed by Henderson-sellers.

1.2 Data Selection

We are using 20 datasets from promise repository, 10 of balanced dataset and 10 of imbalanced dataset.

Table 4.1 summarizes the balanced dataset and imbalanced dataset that we have considered. We have considered moderately to highly imbalanced datasets.

In a dataset each instance is represented by two constituents: a set of 20 OO metrics and a labelled attribute. It is a binary value, 1 if that instance is buggy and 0 if it is clean. Our goal is to predict the instances correctly.

Table 4.1: Summary of the datasets

Dataset	**Version**	**#Classes**	**#Defective**	**Defective %**
Accumulo	1.3.5	679	381	56.11
	1.4.3	1087	111	10.21
Bcel	5.0	563	58	10.3
	5.1	247	144	58.3
Collections	3.1	455	80	17.58
	3.3	340	203	59.7
Jxpath	1.0	125	42	33.6
	1.1	613	499	81.4
Math	1.0	195	34	17.44
	2.1	685	390	56.93
	3.0	619	329	53.15
Pdfbox	1.4.0	596	59	9.9
	1.6.0	645	167	25.89
	1.7.1	195	91	46.7
Poi	1.10	529	88	16.64
	3.0	1515	1185	78.22
	3.7	2472	2088	84.47
	3.9	2786	2377	85.32
Zookeeper	0.01	90	24	26.67
	2.1	107	11	10.28

4.3 Training and Testing Datasets

This section states the training data sets used for a specific testing data set in the study.

We have 10 balanced dataset and 10 imbalanced dataset taken from PROMISE repository. For a balanced testing data, we use all other balanced data except the versions of its own as the training dataset. Similarly, for an imbalanced testing data, we use all other imbalanced data except the versions of its own as the training datasetbecause for CPDP, we use data from other projects for training the model, assuming there is insufficient historical data.Also, data with similar characteristics should be used for training purpose [24][25]. That is why, we are using balanced data as training dataset for testing a balanced data and imbalanced data as training dataset for testing an imbalanced data.

Table 4.2: Training and Testing datasets

Testing Dataset	**Training Dataset**
Accumulo 1.3.5	Bcel 5.1 Collections 3.3 Jxpath 1.1 Math 2.1 Math 3.0 Pdfbox 1.7.1 Poi 3.0 Poi 3.7 Poi 3.9
Accumulo 1.4.3	Bcel 5.0 Collections 3.1 Jxpath 1.0 Math 1.0 Pdfbox 1.4.0 Pdfbox 1.6.0 Poi 1.10 Zookeeper 0.0.1 Zookeeper 2.1
Bcel 5.0	Accumulo 1.4.3 Collections 3.1 Jxpath 1.0 Math 1.0 Pdfbox 1.4.0 Pdfbox 1.6.0 Poi 1.10 Zookeeper 0.0.1 Zookeeper 2.1
Bcel 5.1	Accumulo 1.3.5 Collections 3.3 Jxpath 1.1 Math 2.1 Math 3.0 Pdfbox 1.7.1 Poi 3.0 Poi 3.7 Poi 3.9
Collections 3.1	Accumulo 1.4.3 Bcel 5.0 Jxpath 1.0 Math 1.0 Pdfbox 1.4.0 Pdfbox 1.6.0 Poi 1.10 Zookeeper 0.0.1 Zookeeper 2.1
Collections 3.3	Accumulo 1.3.5 Bcel 5.1 Jxpath 1.1 Math 2.1 Math 3.0 Pdfbox 1.7.1 Poi 3.0 Poi 3.7 Poi 3.9
Jxpath 1.0	Accumulo 1.4.3 Bcel 5.0 Collections 3.1 Math 1.0

	Pdfbox 1.4.0 Zookeeper 2.1	Pdfbox 1.6.0	Poi 1.10	Zookeeper 0.0.1
Jxpath 1.1	Accumulo 1.3.5 Math 3.0 Poi 3.9	Bcel 5.1 Pdfbox 1.7.1	Collections 3.3 Poi 3.0	Math 2.1 Poi 3.7
Math 1.0	Accumulo 1.4.3 Pdfbox 1.4.0 Zookeeper 2.1	Bcel 5.0 Pdfbox 1.6.0	Collections 3.1 Poi 1.10	Jxpath 1.0 Zookeeper 0.0.1
Math 2.1	Accumulo 1.3.5 Pdfbox 1.7.1	Bcel 5.1 Poi 3.0	Collections 3.3 Poi 3.7	Jxpath 1.11 Poi 3.9
Math 3.0	Accumulo 1.3.5 Pdfbox 1.7.1	Bcel 5.1 Poi 3.0	Collections 3.3 Poi 3.7	Jxpath 1.11 Poi 3.9
Pdfbox 1.4.0	Accumulo 1.4.3 Math 1.0	Bcel 5.0 Poi 1.10	Collections 3.1 Zookeeper 0.0.1	Jxpath 1.0 Zookeeper 2.1
Pdfbox 1.6.0	Accumulo 1.4.3 Math 1.0	Bcel 5.0 Poi 1.10	Collections 3.1 Zookeeper 0.0.1	Jxpath 1.0 Zookeeper 2.1
Pdfbox 1.7.1	Accumulo 1.3.5 Math 2.1 Poi 3.9	Bcel 5.1 Math 3.0	Collections 3.3 Poi 3.0	Jxpath 1.11 Poi 3.7
Poi 1.10	Accumulo 1.4.3 Math 1.0 Zookeeper 2.1	Bcel 5.0 Pdfbox 1.4.0	Collections 3.1 Pdfbox 1.6.0	Jxpath 1.0 Zookeeper 0.0.1
Poi 3.0	Accumulo 1.3.5 Math 2.1	Bcel 5.1 Math 3.0	Collections 3.3 Pdfbox 1.7.1	Jxpath 1.11
Poi 3.7	Accumulo 1.3.5 Math 2.1	Bcel 5.1 Math 3.0	Collections 3.3 Pdfbox 1.7.1	Jxpath 1.11
Poi 3.9	Accumulo 1.3.5 Math 2.1	Bcel 5.1 Math 3.0	Collections 3.3 Pdfbox 1.7.1	Jxpath 1.11
Zookeeper 0.0.1	Accumulo 1.4.3 Math 1.0	Bcel 5.0 Pdfbox 1.4.0	Collections 3.1 Pdfbox 1.6.0	Jxpath 1.0 Poi 1.10
Zookeeper 2.1	Accumulo 1.4.3 Math 1.0	Bcel 5.0 Pdfbox 1.4.0	Collections 3.1 Pdfbox 1.6.0	Jxpath 1.0 Poi 1.10

4.4 Performance Measure Selection

Performance of a prediction model is measured by using various performance measures [14, 19, 46]. In this study we are dealing with imbalanced dataset so, performance measures should be chosen carefully. Previous researches show that the traditional performance measures such as accuracy and precision do not provide reliable results in case of imbalanced dataset [55]. A number of studies emphasize the use of robust performance measures for DP using imbalanced dataset. F1-Score and AUC measure are effective measures for estimating the models built with imbalanced dataset. The unequal cost of misclassification errors and skewness in class distributions can be handled by AUC measure robustly. Confusion matrix shows the different performance measures as in Table 4.3.

.

Table 4.3: Confusion Matrix

	Predicted positive	**Predicted negative**
Actual positive(buggy)	True positive(TP)	True negative(TN)
Actual negative(clean)	False positive(FP)	False negative(FN)

- Sensitivity or True Positive Rate (TPR) is the ratio of TP to the total number of defect prone instances.
- Specificity or True Negative Rate (TNR) is the ratio of TN to the total number of defect free instances.
- Precision is used to indicate the number of instances predicted as defect prone and actually contain defects i.e. true positive results.
- F1 score is the harmonic mean of sensitivity and precision. Its value decreases if any one of the factor decreases.
- Are under the ROC curve (AUC) is the plot of sensitivity versus specificity, by taking sensitivity on y-axis and specificity on x-axis.

4.5 Statistical test selection

In this study,the predictive capability of HYDRA using different fitness function on both balanced and imbalanced dataset is evaluated using Wilcoxon test. This test is used to compare two samples and rank the samples according to the absolute values. The various underlying data assumptions, which are necessary for use of parametric tests are not violated as the nature of test is non-parametric [11]. With each performance measure independently,

the predictive performance of the model is compared with the help of Wilcoxon test for evaluating significant differences using different fitness functions.

CHAPTER 5

Result and Analysis

5.1 *Answers to research questions:*

This chapter elaborates the result of our research work and answer different RQs mentioned in chapter one of this thesis.

5.1 RQ1: What is the performance of HYDRA using balanced and imbalanced dataset ?

In order to analyze the performance of HYDRA we are using 20 datasets from promise repository, 10 each of balanced and imbalanced nature. We are analyzing different performance metrics i.e. F1 score and AUC measure, to compare the performance of balanced and imbalanced data on the model. The values in figure 5.1-5.2 represent the median value of performance measures for all the 30 runs executed for HYDRA model. Each bar in the figure represent median value of F1 score and AUC measure for the 20 open source projects. The values corresponding to balanced dataset are depicted using dark grey bars whereas for imbalanced dataset light grey bars are used.

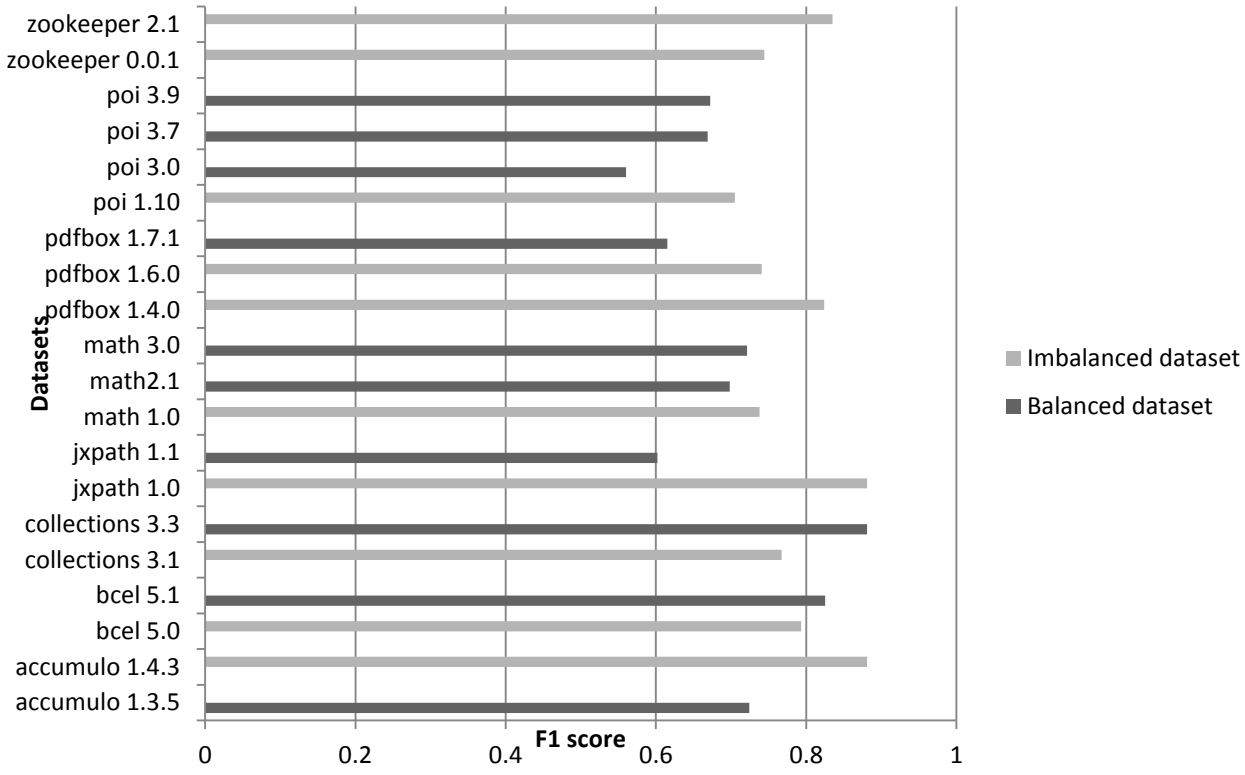

Fig 5.1 Validation Results using Median values of F1 Score for 30 runs

In Fig 5.1, the median value of F1 score ranges from 0.88 to 0.56 for balanced data and from 0.88 to 0.71 for imbalanced data. The best F1 value is for Collections 3.3(balanced) and accumulo 1.4.3(imbalanced) dataset whereas the lowest value is for Poi 3.0(balanced). In case of balanced datasets, 7 out of 10 datasets have F1 score above 0.65. On the other hand, all of the 10 imbalanced datasets show more than 0.65 F1 score. So, in terms of F1 score, it can be seen that HYDRA attains high accuracies on imbalanced datasets.

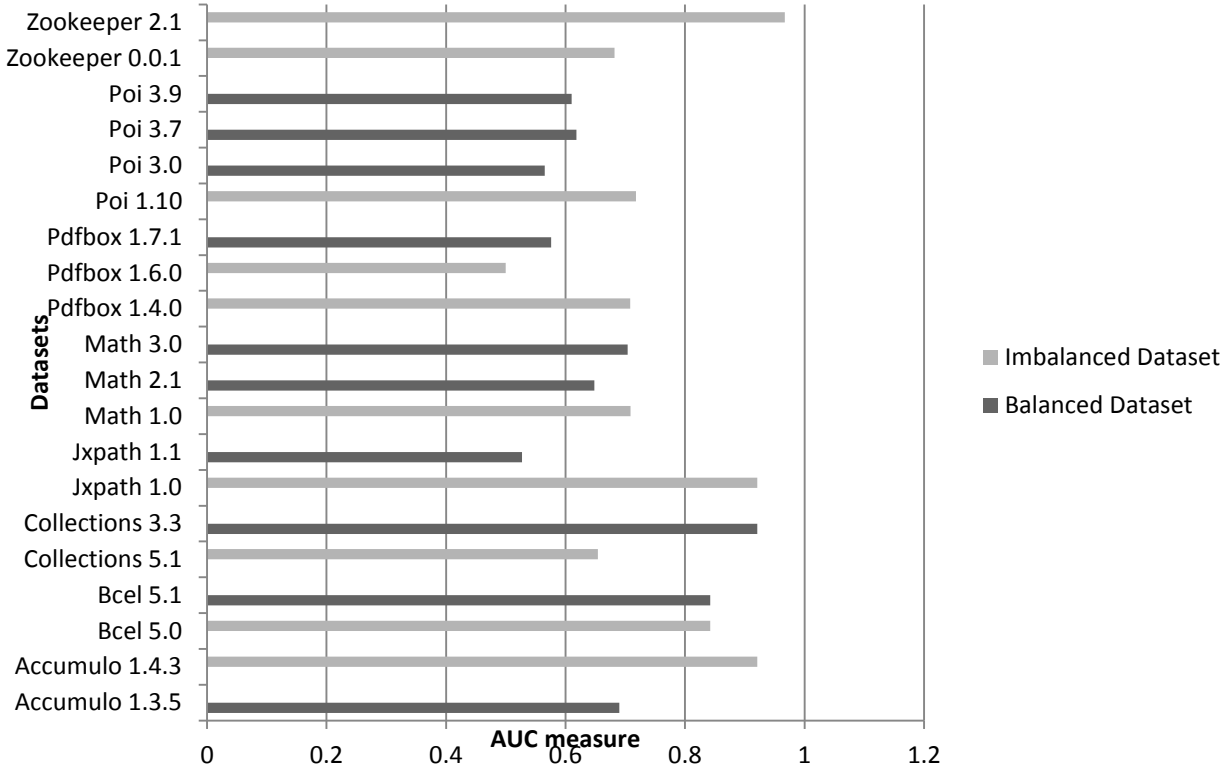

Fig 5.2 Validation Results using Median values of AUC measure for 30 runs

In Fig 5.2, the median value of AUC measure ranges from 0.92 to 0.52 for balanced datasets and from 0.96 to 0.50 for imbalanced datasets. The best value of AUC measure is for Zookeeper 2.1(Imbalanced) and the lowest value is for Pdfbox 1.6.0(imbalanced). In case of balanced data, only 4 out of 10 datasets have more than 0.65 value of AUC measure whereas for imbalanced data, 9 out of 10 datasets have value above than 0.65.Furthermore, it can be seen that in terms of AUC measure, imbalanced dataset has above average performance on HYDRA. The results of Xia et al. [66] on balanced dataset has a range of 0.34 to 0.99 using F1 score. It shows that our result is in accordance with that of Xia et al. [66]. Moreover, our experiment using imbalanced dataset shows that HYDRA has a great accuracy on imbalanced datasets as well.

The performance of HYDRA is acceptable with respect to both balanced as well as imbalanced datasets. The mean AUC measure that is achieved on imbalanced datasets is 0.76 and that of balanced datasets is 0.67. While the mean F1 score is 0.79 on imbalanced datasets and 0.69 on balanced datasets. The result shows the efficiency of HYDRA on both nature of datasets.

RQ2: Does the results of HYDRA vary with change in fitness function?If yes, which fitness function gives better results on the investigated datasets?

We validate whether HYDRA is able to perform better with change in fitness function. The main aim of this research question is to determine, which of the two fitness functions amongst F1 score and AUC measure achieves a better performance of HYDRA model. For this, we analyzed the 20 investigated datasets with two performance metrics i.e. F1 score and AUC measure. The median value of 30 runs is taken for each of the performance metric.

Fig 5.3 shows the comparison of the HYDRA model with two investigated fitness functions using F1 score as performance metric on all the 20 datasets being evaluated. Using F1 score as fitness function, 12 out 20 datasets obtain an F1 score of more than 0.60, while using AUC measure as a fitness function, we have 18 cases out of 20 with more than 0.60 F1 score. Overall, in 13 out of 20 cases, the F1 score value of AUC measure as fitness function outperforms to that of F1 score as fitness function.

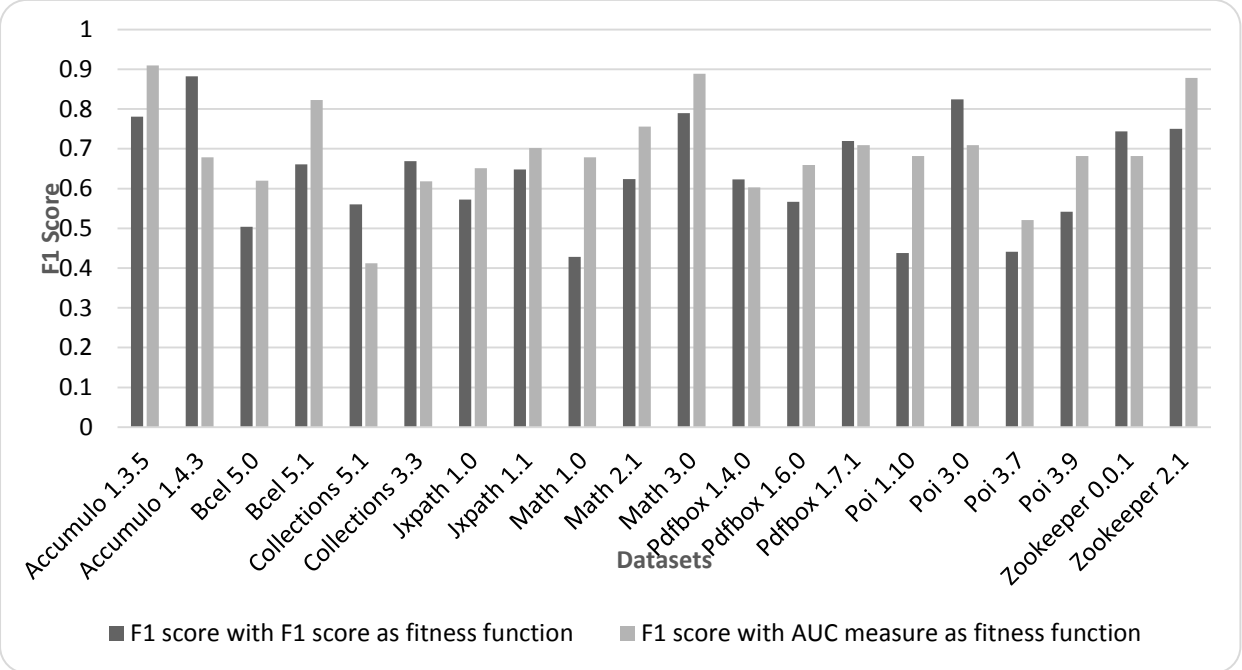

Fig 5.3: Median F1 score results using F1 score and AUC as fitness function for 30 runs

Further, Fig 5.4 depicts the AUC results and compares the results of HYDRA model with the two fitness functions (F1 score and AUC measure). In 13 out of 20 cases, the obtained AUC values were above 0.60 using F1 score as fitness function. While using AUC measure as fitness function, we have 16 out of 20 cases with AUC values more than 0.60. Overall, in 16

out of 20 cases AUC measure as fitness function performs better as compared to F1 score, when evaluated using AUC values.

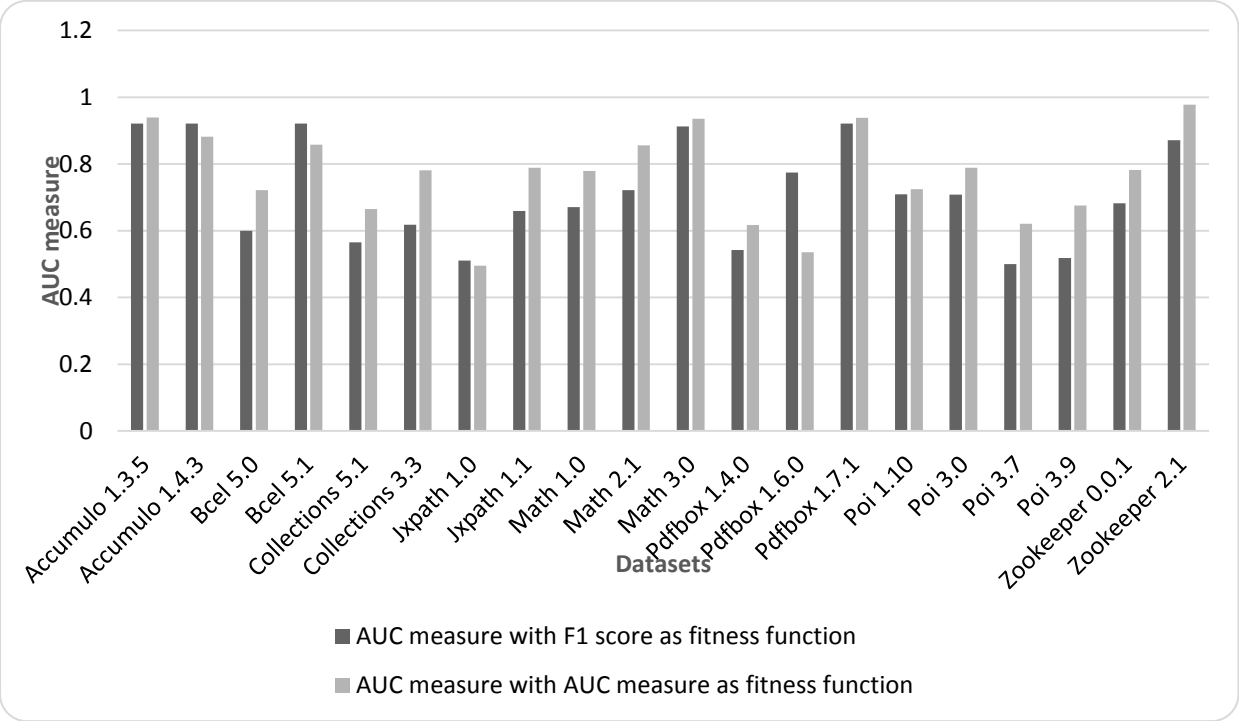

Fig 5.4: Median AUC results using F1 score and AUC as fitness function for 30 runs

In both the experiments of using different performance measures, AUC measure as fitness function provides better results as compared to that of F1 score as fitness function for the HYDRA model. To validate the obtained results statistically, we have used Wilcoxon test. The comparison test is performed at significance level of 0.05 using the performance metrics used in the study i.e. F1 score and AUC measure for all the 20 investigated data sets. The Wilcoxon test results are shown in Table 5.1.

Table 5.1: Wilcoxon Test Result

AUC measure as fitness of HYDRA vs F1 score as fitness of HYDRA	Using F1 score as performance measure	Using AUC measure as performance measure
	S+	S+

S+ -> AUC measure as fitness of HYDRA outperforms F1 score as fitness of HYDRA significantly.

According to table 5.1, the AUC measure as fitness function significantly outperforms F1 score as fitness function, when the results were evaluated using both F1 score and AUC measure. Thus, the results of Wilcoxon test advocate that AUC measure as fitness function is a better fitness function that improves the accuracy of model HYDRA for DP.

The results of HYDRA do vary with change in fitness function. The F1 score of the investigated datasets using AUC measure as fitness function lies in the range of 0.91 to 0.41 while using F1 score as fitness function the range is 0.88 to 0.42. The AUC measure using AUC measure as fitness function has the range 0.97 to 0.535 while using F1 score as fitness function the range is 0.92 to 0.50. From the results,it can be seen that AUC measure as fitness function outperforms F1 score as fitness function. Wilcoxon test is used to statistically verify the results.

CHAPTER 6

Conclusion and Future Work

6.1 *Conclusion*

The study evaluates the use of imbalanced datasets for CPDP models. The main objective is to analyze the performance of HYDRA, a CPDP model, on balanced as well as imbalanced datasets. Furthermore, the study analyzes the variation in the performance of HYDRA by using different fitness functions i.e. F1 score and AUC measure. The experiments are performed using 30 independent runs on twenty open source datasets. The results are assessed using AUC values and F1 score. Wilcoxon test is used to compare the performance of HYDRA using F1 score and AUC measure as fitness functions.

The results of the study are summarized as follows.

- In our study, we validated the use of imbalanced datasets on HYDRA. The results verified that HYDRA can successfully predict defective classes on imbalanced datasets with good accuracies. The F1 score obtained by using imbalanced datasets on HYDRA is in the range of 0.88 to 0.71 whereas, the AUC results are in the range of 0.96 to 0.50. In case of balanced dataset, F1 score provides the results in the range of 0.88 to 0.56,whilethe AUC measure varies from 0.92 to 0.52. Thus, HYDRA is effective for both balanced as well as imbalanced data sets.

- We verified variation in the performance of HYDRA with change in its fitness function. In majority of the datasets, AUC measure performs better than F1 score as fitness function. The mean value of F1 score increased by 8.1% and the mean value of AUC measure is increased by 7.7% by using AUC measure as fitness function over F1 score as fitness function. Wilcoxon test verifies that the improvement in performance of HYDRA using AUC measure as fitness function is significant over F1 score as fitness function.

Thus, the performance of model HYDRA used in our study shows that imbalanced dataset can also provide promising results on CPDP models and a model can be effectively used to predict defective classes in such datasets. Also, the study improves the performance of model HYDRA by changing its fitness function to AUC measure. The outcomes of this study can be efficiently used by researchers and software developers while handling cases where the training data for a CPDP model is of imbalanced nature. Moreover, the HYDRA model has also been improved by changing its fitness function so, it can be efficiently used to detect the defect prone classes with high accuracy. With this study, we provide a very efficient model for CPDP that performs equally well on both balanced and imbalanced dataset.

6.2 Future Work

As a part of the future work, the datasets with programming language other than java can be used for empirical validation of the HYDRA model, so that the results can be evaluated and generalized for other programming languages. Moreover, we can analyze the use of multi-objective genetic algorithm instead of basic genetic algorithm in HYDRA model. Also, the use of ensemble learning along with genetic algorithm can be evaluated to develop CPDP models.

Reference

1. Aslam, M. W. (2015, July). Selection of fitness function in genetic programming for binary classification. In *Science and Information Conference (SAI), 2015* (pp. 489-493). IEEE.

2. Bansiya, J., & Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on software engineering*, *28*(1), 4-17.

3. Basili, V. R., Briand, L. C., & Melo, W. L. (1996). A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on software engineering*, *22*(10), 751-761.

4. Bekkar, M., Djemaa, H. K., &Alitouche, T. A. (2013). Evaluation measures for models assessment over imbalanced data sets. *Iournal Of Information Engineering and Applications*, *3*(10).

5. Bhowan, U., Johnston, M., & Zhang, M. (2012). Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*(2), 406-421.

6. Camargo Cruz, A. E., &Ochimizu, K. (2009, October). Towards logistic regression models for predicting fault-prone code across software projects. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement* (pp. 460-463). IEEE Computer Society.

7. Catal, C., &Diri, B. (2009). Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences*, *179*(8), 1040-1058.

8. Chidamber, S. R., &Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on software engineering*, *20*(6), 476-493.

9. D'Ambros, M., Lanza, M., &Robbes, R. (2010, May). An extensive comparison of bug prediction approaches. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on* (pp. 31-41). IEEE.

10. De Carvalho, A. B., Pozo, A., &Vergilio, S. R. (2010). A symbolic fault-prediction model based on multiobjective particle swarm optimization. *Journal of Systems and Software*, *83*(5), 868-882.

11. Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, *7*(Jan), 1-30.

12. Di Martino, S., Ferrucci, F., Gravino, C., &Sarro, F. (2011, June). A genetic algorithm to configure support vector machines for predicting fault-prone components. In *International Conference on Product Focused Software Process Improvement* (pp. 247-261). Springer, Berlin, Heidelberg.

13. El Emam, K., Melo, W., & Machado, J. C. (2001). The prediction of faulty classes using object-oriented design metrics. *Journal of Systems and Software*, *56*(1), 63-75.

14. Elish, K. O., &Elish, M. O. (2008). Predicting defect-prone software modules using support vector machines. *Journal of Systems and Software*, *81*(5), 649-660.

15. FernáNdez, A., LóPez, V., Galar, M., Del Jesus, M. J., & Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-based systems*, *42*, 97-110.

16. Ferrucci, F., Gravino, C., Oliveto, R., &Sarro, F. (2010, September). Genetic programming for effort estimation: an analysis of the impact of different fitness functions. In *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on* (pp. 89-98). IEEE.

17. Gao K, Khoshgoftaar TM, Napolitano A (2015) Combining feature subset selection and data sampling for coping with highly imbalanced software data. In Proc. of 27th International Conf. on Software Engineering and Knowledge Engineering, Pittsburgh, 2015

18. Gondra, I. (2008). Applying machine learning to software fault-proneness prediction. *Journal of Systems and Software*, *81*(2), 186-195.

19. Guo, L., Ma, Y., Cukic, B., & Singh, H. (2004, November). Robust prediction of fault-proneness by random forests. In *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on* (pp. 417-428). IEEE.

20. Gyimothy, T., Ferenc, R., &Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software engineering*, *31*(10), 897-910.

21. He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, *21*(9), 1263-1284.

22. He, Z., Shu, F., Yang, Y., Li, M., & Wang, Q. (2012). An investigation on the feasibility of cross-project defect prediction. *Automated Software Engineering*, *19*(2), 167-199.

23. Henderson-Sellers, B. (1995). *Object-oriented metrics: measures of complexity*. Prentice-Hall, Inc..

24. Herbold, S. (2013, October). Training data selection for cross-project defect prediction. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering* (p. 6). ACM.

25. Hosseini, S., Turhan, B., &Mäntylä, M. (2016, September). Search Based Training Data Selection For Cross Project Defect Prediction. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering* (p. 3). ACM.

26. Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013, September). Facing imbalanced data--Recommendations for the use of performance metrics. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on* (pp. 245-251). IEEE.

27. Jing, X. Y., Wu, F., Dong, X., & Xu, B. (2017). An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems. *IEEE Transactions on Software Engineering*, *43*(4), 321-339.

28. Jureczko, M., &Madeyski, L. (2010, September). Towards identifying software project clusters with regard to defect prediction. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering* (p. 9). ACM.

29. Jureczko, M., &Spinellis, D. (2010). Using object-oriented design metrics to predict software defects. *Models and Methods of System Dependability. OficynaWydawniczaPolitechnikiWrocławskiej*, 69-81.

30. Kamei, Y., Fukushima, T., McIntosh, S., Yamashita, K., Ubayashi, N., & Hassan, A. E. (2016). Studying just-in-time defect prediction using cross-project models. *Empirical Software Engineering*, *21*(5), 2072-2106.

31. Kanmani, S., Uthariaraj, V. R., Sankaranarayanan, V., &Thambidurai, P. (2007). Object-oriented software fault prediction using neural networks. *Information and software technology*, *49*(5), 483-492.

32. Khoshgoftaar, T. M., Seliya, N., & Drown, D. J. (2010). Evolutionary data analysis for the class imbalance problem. *Intelligent Data Analysis*, *14*(1), 69-88.

33. Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, *34*(4), 485-496.

34. Li, M., Zhang, H., Wu, R., & Zhou, Z. H. (2012). Sample-based software defect prediction with active and semi-supervised learning. *Automated Software Engineering*, *19*(2), 201-230.

35. Liu, Y., An, A., & Huang, X. (2006, April). Boosting Prediction Accuracy on Imbalanced Datasets with SVM Ensembles. In *PAKDD* (Vol. 6, pp. 107-118).

36. Longadge, R., &Dongre, S. (2013). Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707*.

37. López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*, 113-141.

38. Ma, Y., Luo, G., Zeng, X., & Chen, A. (2012). Transfer learning for cross-company software defect prediction. *Information and Software Technology*, *54*(3), 248-256.

39. Mahmood, Z., Bowes, D., Lane, P. C., & Hall, T. (2015, October). What is the Impact of Imbalance on Software Defect Prediction Performance?. In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering* (p. 4). ACM.

40. Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, *27*, 504-518.

41. Malhotra, R., & Singh, Y. (2011). On the applicability of machine learning techniques for object oriented software fault prediction. *Software Engineering: An International Journal*, *1*(1), 24-37.

42. Malhotra, R., Shukla, S., & Sawhney, G. (2016, September). Assessment of defect prediction models using machine learning techniques for object-oriented systems. In *Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2016 5th International Conference on* (pp. 577-583). IEEE.

43. Martin, R. (1994). OO design quality metrics. *An analysis of dependencies*, *12*, 151-170.

44. McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on software Engineering*, (4), 308-320.

45. Misirli, A. T., Bener, A., & Kale, R. (2011). Ai-based software defect predictors: Applications and benefits in a case study. *AI Magazine*, *32*(2), 57-68.

46. Nagappan, N., Ball, T., & Zeller, A. (2006, May). Mining metrics to predict component failures. In *Proceedings of the 28th international conference on Software engineering* (pp. 452-461). ACM.

47. Özturk, M. M., &Zengin, A. (2016, September). HSDD: A hybrid sampling strategy for class imbalance in defect prediction data sets. In *Digital Information Management (ICDIM), 2016 Eleventh International Conference on* (pp. 225-234). IEEE.

48. Pai, G. J., & Dugan, J. B. (2007). Empirical analysis of software fault content and fault proneness using Bayesian methods. *IEEE Transactions on software Engineering*, *33*(10).

49. Panichella, A., Oliveto, R., & De Lucia, A. (2014, February). Cross-project defect prediction models: L'union fait la force. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on* (pp. 164-173). IEEE.

50. Qing, H., Biwen, L., Beijun, S., & Xia, Y. (2015, November). Cross-Project Software Defect Prediction Using Feature-Based Transfer Learning. In *Proceedings of the 7th Asia-Pacific Symposium on Internetware* (pp. 74-82). ACM.

51. Rodriguez, D., Herraiz, I., Harrison, R., Dolado, J., &Riquelme, J. C. (2014, May). Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (p. 43). ACM.

52. Ryu, D., &Baik, J. (2016). Effective multi-objective naïve bayes learning for cross-project defect prediction. *Applied Soft Computing*, *49*, 1062-1077.

53. Ryu, D., Jang, J. I., &Baik, J. (2015). A hybrid instance selection using nearest-neighbor for cross-project defect prediction. *Journal of Computer Science and Technology*, *30*(5), 969-980.

54. Ryu, D., Jang, J. I., &Baik, J. (2017). A transfer cost-sensitive boosting approach for cross-project defect prediction. *Software Quality Journal*, *25*(1), 235-272.

55. Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., &Folleco, A. (2014). An empirical study of the classification performance of learners on imbalanced and noisy software quality data. *Information Sciences*, *259*, 571-595.

56. Sun, Y., Wong, A. K., &Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(04), 687-719.

57. Tang, M. H., Kao, M. H., & Chen, M. H. (1999). An empirical study on object-oriented metrics. In *Software Metrics Symposium, 1999. Proceedings. Sixth International* (pp. 242-249). IEEE.

58. Turhan, B., Bener, A., & Menzies, T. (2010, June). Regularities in learning defect predictors. In *International Conference on Product Focused Software Process Improvement* (pp. 116-130). Springer, Berlin, Heidelberg.

59. Turhan, B., jurecz, T., Bener, A. B., & Di Stefano, J. (2009). On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, *14*(5), 540-578.

60. Wang, B. X., &Japkowicz, N. (2010). Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, *25*(1), 1-20.

61. Wang, S., & Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, *62*(2), 434-443.

62. Watanabe, S., Kaiya, H., &Kaijiri, K. (2008, May). Adapting a fault prediction model to allow inter languagereuse. In *Proceedings of the 4th international workshop on Predictor models in software engineering* (pp. 19-24). ACM.

63. Weyuker, E. J., Ostrand, T. J., & Bell, R. M. (2008). Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models. *Empirical Software Engineering*, *13*(5), 539-559.

64. Weyuker, E. J., Ostrand, T. J., & Bell, R. M. (2010). Comparing the effectiveness of several modeling methods for fault prediction. *Empirical Software Engineering*, *15*(3), 277-295.

65. Xia, X., Lo, D., Pan, S. J., Nagappan, N., & Wang, X. (2016). Hydra: Massively compositional model for cross-project defect prediction. *IEEE Transactions on software Engineering*, *42*(10), 977-998.

66. Yu, Q., Jiang, S., & Zhang, Y. (2017). The Performance Stability of Defect Prediction Models with Class Imbalance: An Empirical Study. *IEICE TRANSACTIONS on Information and Systems*, *100*(2), 265-272.

67. Zhang, F., Zheng, Q., Zou, Y., & Hassan, A. E. (2016, May). Cross-project defect prediction using a connectivity-based unsupervised classifier. In *Proceedings of the 38th International Conference on Software Engineering* (pp. 309-320). ACM.

68. Zimmermann, T., Nagappan, N., Gall, H., Giger, E., & Murphy, B. (2009, August). Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering* (pp. 91-100). ACM.