COMPARISION OF ENSEMBLE LEARNING MODELS AND IMPACT OF DATA BALANCING TECHNIQUE FOR SOFTWARE EFFORT ESTIMATION

A PROJECT REPORT

SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF DEGREE
OF

MASTER OF TECHNOLOGY
IN
SOFTWARE ENGINEERING

Submitted By

Misha Jawa
(2K20/SWE/14)

Under the supervision of

Ms. Shweta Meena
(Assistant Professor)

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

May, 2022

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

DECLARATION

I, Misha Jawa, 2K20/SWE/14 student of M.Tech (SWE), hereby declare that the project entitled " Comparison of Ensemble Learning Models and Impact of Data Balancing Technique for Software Effort estimation " which is submitted by me to Department of Software Engineering, Delhi Technological University, Delhi in partial fulfilment of requirement for the award of the degree of Master of Technology in Software Engineering, is original and not copied from any source without proper citation. This work has not previously formed for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Misha

Place: Delhi

Misha Jawa

Date: May, 2022

(2K20/SWE/14)

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the project entitled **"Comparison of Ensemble Learning Models and Impact of Data Balancing Technique for Software Effort estimation"** which is submitted by Misha Jawa (2K20/SWE/14) to Department of Software Engineering, Delhi Technological University, Delhi in partial fulfilment of requirement for the award of the degree of Master of Technology in Software Engineering, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this university or elsewhere.

Shweta
26/8/2022

Place: Delhi
Date: May, 2022

Ms. Shweta Meena
Assistant Professor
Department of SWE

ACKNOWLEDGEMENT

I am very thankful to Ms. Shweta Meena (Assistant Professor, Department of Software Engineering) and all the faculty members of the Department of Software Engineering at DTU. They all provided us with immense support and guidance for the project. I would also like to express my gratitude to the University for providing us with the laboratories, infrastructure, testing facilities and environment which allowed us to work without any obstructions. I would also like to appreciate the support provided to us by our lab assistants, seniors and our peer group who aided us with all the knowledge they had regarding various topics.

Misha

Misha Jawa

2K20/SWE/14

ABSTRACT

Project management is a critical component of every software project's success. Estimating the cost and effort of software development at the outset of the project is one of the most important responsibilities in software project management. Estimating effort allows project managers to more effectively manage resources and activities. The primary purpose of this study was to construct and compare the usage of two common ensemble approaches (bagging and boosting) to improve estimator accuracy and to study the impact of Synthetic Minority Over-Sampling Technique for Regression (SMOTER) to predict effort estimation by using machine learning algorithms.
Random forest, support vector regression, elastic net, decision tree regressor, linear regression, lasso regression, and ridge regression are some of the machine learning techniques we've implemented. For our study we used Albrecht, China, COCOMO81, Desharnais and Maxwell dataset. We also performed feature selection and considered only those features that have strong correlation with target feature i.e., effort.

The two-performance metrics Mean Magnitude Relative Error (MMRE) and PRED(25) results demonstrate that utilising elastic net as the base learner for AdaBoost outperforms the other models and there is a significant decrease in error of each model after applying SMOTER.

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Project management is an essential component of every software project's success. Estimating the cost and effort of software development at the outset of the project is one of the most vital responsibilities in project management. Project managers can better manage resources and activities by estimating effort. The majority of this software development cost is made up of human effort, and it gives estimates in person-months. Effort estimation is used to determine how many months the programme will take to build and how many resources will be required to accomplish the project on time. Both clients and developers benefit greatly from realistic cost estimates. It can be used to generate contract negotiations, proposals, monitoring, scheduling, and control requests. Management may approve proposed systems that exceed allowable budgets, have underdeveloped functionalities and poor quality, and fail to execute on time due to major cost underestimation. Overestimating software projects can result in the project receiving more assets or the contract being lost during contract bidding, resulting in job loss.

Despite the fact that Software Effort Estimation (SEE) is critical in software project development, there has been relatively little progress in the last thirty or forty years. One of the most common causes of failure is inaccurate resource estimates. Even though there are numerous software effort estimation models available, new models are still needed to improve estimation accuracy since the problem of effort estimation and accuracy remains the same. As a result of the emergence of software effort prediction models, researchers are now attempting to estimate software work as precisely as possible.

Cost estimation accuracy is crucial because:

- It is used to identify the resources to be used for the project and how well these assets will be utilized.
- Impacts of changes are assessed and re-planning is facilitated.

- Control and management of project is easier when resources are compatible to actual needs.
- Customers anticipate the least amount of variance between real and estimated development expenses.

1.1 Motivation

The biggest issue that data scientists face nowadays is data imbalance. The class imbalance problem in machine learning relates to classification jobs in which data classes are not represented equally. In many real-world applications, the nature of the task necessitates a skewed in the classification process of a binary or multi-class classification problem [1]. This problem is also faced in dataset when target value is continuous which is known as imbalanced regression. In reality, the continuous structure of the target variable in regression data sets complicates the process because there are potentially an unlimited number of values to deal with [2]. To overcome this issue of imbalance data, data oversampling is done. Data oversampling is a technique for producing data that closely reflects the underlying distribution of real data. Synthetic Minority Over-Sampling Technique (SMOTE) is an oversampling technique for balancing datasets [3].

To model the intricate link between effort and software properties, machine learning techniques are frequently used. Researchers have used a variety of machine learning-based estimators such as Regression Trees (RT), Support Vector Regression (SVR), and ensemble models like random forest, Artificial Neural Network (ANN), Multilayer Perceptron (MLP), Generalized Linear Model (GLM) [4] [5] [6]. Ensemble learning approaches have recently gained popularity. The results depicts that in the vast large majority of cases, Ensemble Effort Estimation (EEE) methods outperform single models [7]. There have been several different approaches for estimating software expenses, or SEE, but there have been very few studies on hyperparameter tuning.

Underestimating the software effort or resource requirements for a software project can result in an unrealistic timeline, cost, and manpower estimates. It also puts more strain on the engineers' workload. Overestimating the work, on the other side, is likely to result in low engineer productivity and a contract loss due to prohibitive prices.

1.2 Research Questions

This thesis focused on mainly these research questions:

RQ 1: To check whether ensemble techniques improve the prediction performance as compared to individual regression techniques used or not?
RQ2: Does hyperparameter tuning enhance the results for software effort prediction?
RQ3: Which is the best ensemble technique for software effort prediction?
RQ4: Does data balancing improves the result of prediction?

1.3 Thesis Structure

The main purpose of this thesis is to compare the accuracy of numerous ensemble learning algorithms for estimating software effort. The results obtained from our studies will help to improve the prediction in future. To achieve this, we have used two ensembles learning i.e., AdaBoost and bagging. To improve the results, we have also tuned the 'base_learner' hyperparameter with several weak learners such as SVR, random forest and elastic net. These techniques are the modern trends in the field of effort estimation. To evaluate the results three performance metrics MMRE, MdMRE and PRED(25) have been considered. For results, these techniques are applied on Albrecht, China and Desharnais dataset consisting of 24, 499 and 81 projects respectably.

The rest of the thesis is organized into varied chapters. In chapter 2, related work done for the research have been explained in brief. In chapter 3, all the terminologies used are described which includes datasets, Models of machine learning that have been utilised as the base learners in our ensemble models and the performance metrics. In chapter 4 the research methodology which describes the flow of our research is explained. In chapter 5 the results after application of different ensemble techniques have been explained. The impact of data balancing techniques on prediction has also been discussed. Conclusion and future work have been explained in chapter 6 and chapter 7 respectively. Finally, all the references used in out thesis have been mentioned.

CHAPTER 2

LITERATURE REVIEW

In calculating the development cost of a software project, software effort estimation is critical. In a prior study, many methodologies such as empirical techniques, algorithmic effort estimation, regression techniques, theory-based techniques, and machine learning were provided, as well as many models. In software project management, recognising and controlling crucial variables that affect development effort is critical.

The accuracy of our data prediction is mostly determined by two factors: first, the models we use, and second, the dataset. We must choose models that will produce the best results for the type of data we are predicting, whether it is a classification or continuous data. The dataset we're evaluating should be of high quality, which implies it should be balanced. Because obtaining a balanced dataset is challenging, data oversampling is used to achieve that goal.

The large majority of machine learning regression models are based on data that is balanced or roughly balanced. When dealing with unbalanced data, such models will be misleading and perform badly. We have two alternatives for dealing with imbalanced data using these models: increase the representation of the interest observations vs. the other observations, or lower the representation of the interest observations vs. the other observations [8]. The SMOTE algorithm employs an oversampling method to rebalance the original training set. SMOTE's core concept is to create instances rather than simply replicating minority class occurrences. Interpolating across various minority class instances within a single neighbourhood in order to produces new data [9]. This problem also exists in datasets with regression as the target value, and SMOTER is employed to solve it.

In early 20’s machine learning models had an edge in field of prediction, machine learning techniques were found in numerous applications like link prediction [10] , fault prediction [11], customer churn prediction [12], software effort prediction [4] however, it has recently been supplanted by ensemble approaches, which integrate the judgments of numerous models to increase overall performance. Ensemble approach comprises Bagging, which incorporates the outputs of many models to produce a generalised result, boosting, which aims to overcome the weaknesses of the earlier models, and stacking, which merges the predictions of various weak learners using the soft or hard voting concept, are all examples of the ensemble approach.

A. Priya Varshini, in her paper tried to address SEE by using different machine learning techniques namely all types of regression models, random forest and Neural Network (NN) to recognize best performing model. Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and R-Squared performance metrics were considered for evaluation. Support Vector Machine (SVM) outperform other machine learning methods in her comparison analysis [13].

The Genetic Algorithm (GA) has been widely utilised to estimate effort accurately. According to Burgess and Lefley [14], the ability of "Genetic Programming" has been employed for effort estimation and comparing them, with the ANN, Linear LSR, and other methods. The comparison is made using the Desharnais dataset, and the findings are completely dependent on the fitness function utilised.

On the basis of machine learning, Braga et al. [15] suggested an approach that provides effort estimation and confidence interval. The authors advise that substantial confidence intervals be used. This confidence range is independent of the error probability distribution in the training set. Several experiments have been conducted with two software project datasets: "Desharnais and NAS," which are expected to investigate machine learning algorithms for reliable effort estimation. Following simulations, the proposed strategy was able to generate confidence intervals, which are crucial for a client of the effort estimating system. Martin et al. Lopez-Martin [16] in 2006 proposed a novel methodology for estimating software development effort and offered an updated Fuzzy Logic Model for measuring it. Elish [17] conducted a comparative study to develop software effort prediction systems using machine learning approaches such as ANNs, CBR, and RI (Rule Induction).

S. K. Palaniswamy, provide a method for increasing the accuracy of effort estimation utilizing the stacking ensemble method and evolutionary algorithms to tune the hyperparameters of the base and meta learners. They utilized four distinct base learners to improve the accuracy of this stacking ensemble method: Linear Regression (LR), MLP, Random Forest Regressor (RFR), and AdaBoost regressor. Based on their findings, they believe that a combination of hyperparameter configurations can produce close to optimal configuration setting values [18].

The paper by Bibi and Stamelos [19], has suggested a set of estimation criteria for choosing on an acceptable machine learning technique for software development effort estimation in terms of comprehensibility, causality, accuracy, handling of missing information, dynamic updates, uncertainty, sensitivity, and application. As a result, the author suggested various principles for selecting an acceptable estimation method based on his needs and for the estimator's research. The paper by Gallego et al. [20], the author has demonstrated one of the most significant issues facing developers in terms of predicting software development effort regarding project size, complexity, developer competencies, and other variables.

In the estimation of effort, machine learning (ML) approaches have been widely applied. Finnie and Witting's [21] [22] study looked at the ability of two artificial intelligence (AI) approaches, ANN and "Case Based Reasoning" (CBR), to improve application evaluation models. Furthermore, rather than relapse models, the capability of ANN and CBR allows for improvement in application estimating models. AI models are designed to provide adequate evaluation models. Both models' performance is substantially determined by training data and to the extent to which data is available. ML techniques such as "Classification and Regression Trees" (CART) have been published in addition to ANN and CBR [23]. MART as a model of software effort estimation was compared to previously existing models such as SVR models with linear regression, RBF kernels, RBF neural networks, and linear regression in Elish's [17]study. The accuracy of the final effort estimation with MART is higher.

Pendharkar [24] proposed a "Probabilistic Neural Networks" (PNN) methodology for simultaneous estimation of the values of development parameters, such as effort or size, as well as the chance that the estimated value of the attribute will be greater than the actual value in his study. According to Radlinski and Hoffmann [25] , the researcher compared the accuracy of software effort estimations using multiple machine learning algorithms. The major purpose here is to examine if particular strategies yield the same level of accuracy for diverse data sets to investigate the predictability of this result. The prediction accuracy results reveal that each ML technique uses a different dataset.

Suherman [26], in their study tune the hyperparameters of random forest and compared the results with SVR and Bee Colony Method. The error rate of an experiment is used to evaluate its outcomes. Random forest regression outperforms SVR and the Bee Colony method, according to the results, the results were measured on the basis of MMRE. Dejaeger, K. [27] done comparative study on using data mining techniques in SEE. The key finding in their study was that picking a subset of highly predictive qualities can aid in a large boost in estimation accuracy, and that data mining techniques can add value to the variety of software effort estimating methodologies, but they should not be used in place of expert judgement.

Estimating effort is critical in determining the cost of software. Machine learning models, algorithmic, empirical, and theory-based effort evaluation techniques are among the techniques accessible. Many research initiatives have incorporated machine learning techniques. Machine learning algorithms have been proposed to estimate effort in recent years like Bayesian Network, Case based reasoning, fuzzy logic, GA, SVM, regression tree including deep learning technique ANN have been proposed. We have carried out comparative study of several ensemble models like AdaBoost, bagging by tuning their hyperparameters with machine learning models like random forest, SVR, elastic net for predicting the software effort on different datasets.

CHAPTER 3

TERMONOLOGIES USED

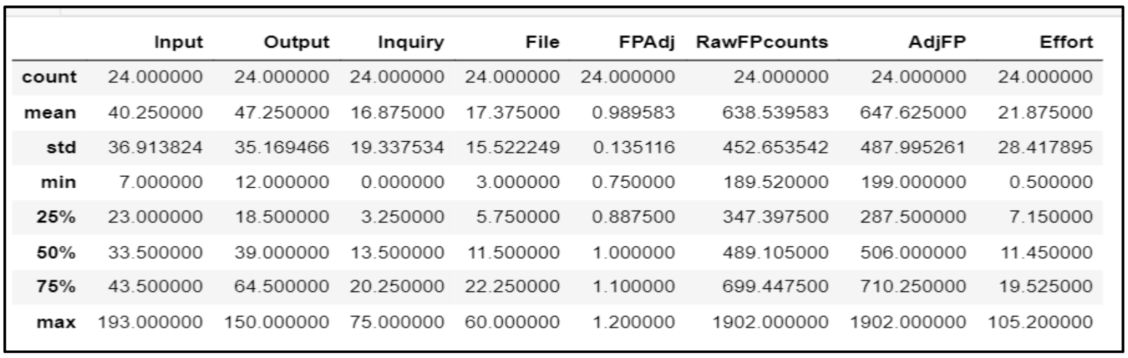
There are five varied length datasets that we have selected for our research namely Albrecht, China, COCOMO81, Desharnais and Maxwell dataset. Considering the shift in research from machine learning models to ensemble models, we opted to focus on two ensemble models: bagging and AdaBoost. We further tune the hyperparameter of ensemble models with random forest, SVR and elastic net. To study the impact of the data balancing we used over-sampling technique SMOTER on five popular machine learning models random forest, decision tree regressor, linear regression, lasso regression and ridge regression.

3.1 Dataset Used

For our studies we have used five freely available datasets namely Albrecht dataset [28], China dataset [29], COCOMO81 dataset [30], Desharnais dataset [31] and Maxwell dataset [32].

3.1.1 Albrecht Dataset

The Albrecht dataset was extracted from Zenodo repository. It has eight features and consists of 24 software projects data.



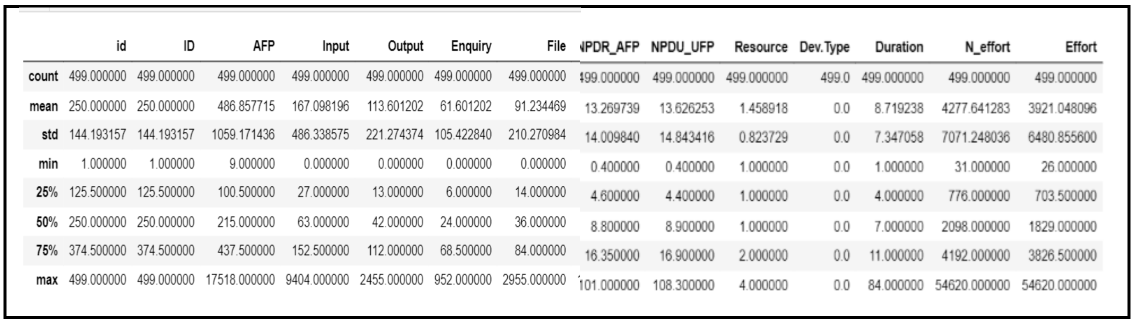
	Input	Output	Inquiry	File	FPAdj	RawFPcounts	AdjFP	Effort
count	24.000000	24.000000	24.000000	24.000000	24.000000	24.000000	24.000000	24.000000
mean	40.250000	47.250000	16.875000	17.375000	0.989583	638.539583	647.625000	21.875000
std	36.913824	35.169466	19.337534	15.522249	0.135116	452.653542	487.995261	28.417895
min	7.000000	12.000000	0.000000	3.000000	0.750000	189.520000	199.000000	0.500000
25%	23.000000	18.500000	3.250000	5.750000	0.887500	347.397500	287.500000	7.150000
50%	33.500000	39.000000	13.500000	11.500000	1.000000	489.105000	506.000000	11.450000
75%	43.500000	64.500000	20.250000	22.250000	1.100000	699.447500	710.250000	19.525000
max	193.000000	150.000000	75.000000	60.000000	1.200000	1902.000000	1902.000000	105.200000

Fig 3.1 Albrecht dataset description

There are seven independent variables namely ‘Input’, ‘output’, ‘Inquiry’, ‘File’, ‘FPAdj’, ‘RawFPcounts’, ‘AdjFP’ and one dependent variable ‘Effort’ whose values rages from 0.5 to 105.2 person-month. The six features with high correlation with target feature are selected for further research and remaining were droped.

3.1.2 China Dataset

The China dataset was extracted from Zenodo repository. It consists of 499 software projects data described by 19 features.



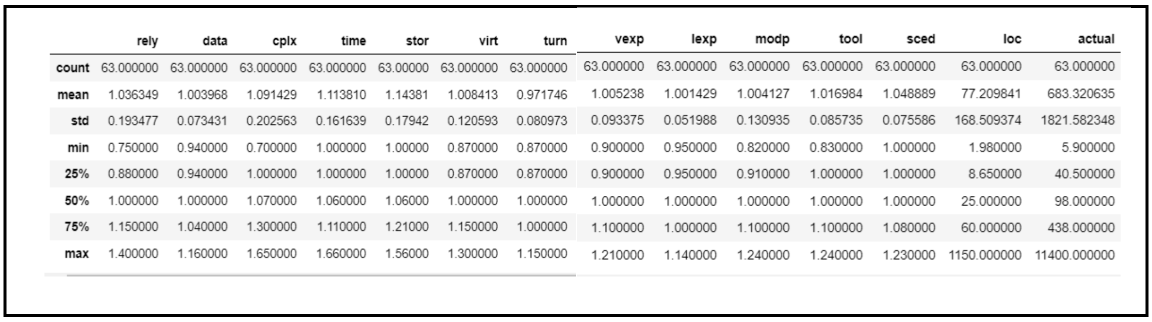
	id	ID	AFP	Input	Output	Enquiry	File	NPDR_AFP	NPDU_UFP	Resource	Dev.Type	Duration	N_effort	Effort
count	499.000000	499.000000	499.000000	499.000000	499.000000	499.000000	499.000000	499.000000	499.000000	499.000000	499.0	499.000000	499.000000	499.000000
mean	250.000000	250.000000	486.857715	167.098196	113.601202	61.601202	91.234469	13.269739	13.626253	1.458918	0.0	8.719238	4277.641283	3921.048096
std	144.193157	144.193157	1059.171436	486.338575	221.274374	105.422840	210.270984	14.009840	14.843416	0.823729	0.0	7.347058	7071.248036	6480.855600
min	1.000000	1.000000	9.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.400000	1.000000	0.0	1.000000	31.000000	26.000000
25%	125.500000	125.500000	100.500000	27.000000	13.000000	6.000000	14.000000	4.600000	4.400000	1.000000	0.0	4.000000	776.000000	703.500000
50%	250.000000	250.000000	215.000000	63.000000	42.000000	24.000000	36.000000	8.800000	8.900000	1.000000	0.0	7.000000	2098.000000	1829.000000
75%	374.500000	374.500000	437.500000	152.500000	112.000000	68.500000	84.000000	16.350000	16.900000	2.000000	0.0	11.000000	4192.000000	3826.500000
max	499.000000	499.000000	17518.000000	9404.000000	2455.000000	952.000000	2955.000000	101.000000	108.300000	4.000000	0.0	84.000000	54620.000000	54620.000000

Fig 3.2 China dataset description

There are eighteen independent features and one dependent feature i.e., ‘effort’. In order to get the features that have strong relation with the target feature i.e., ‘Effort’, feature selection was applied. We have selected seven features ‘AFP’, ‘Input’, ‘Output’, ‘Enquiry’, ‘File’, ‘Added’, ‘N_effort’ and dropped the rest.

3.1.3 COCOMO Dataset

There are 63 software projects in the COCOMO data set, comprising corporate, research, and system projects, that are described by 19 features.



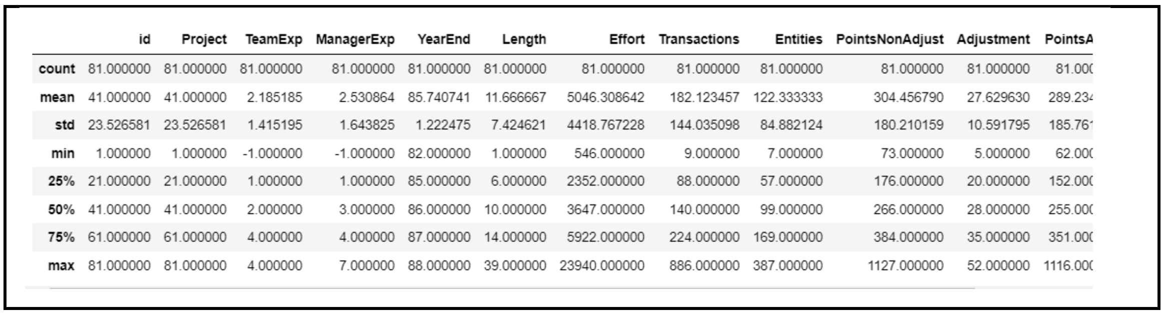
	rely	data	cplx	time	stor	virt	turn	vexp	lexp	modp	tool	sced	loc	actual
count	63.000000	63.000000	63.000000	63.000000	63.00000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000
mean	1.036349	1.003968	1.091429	1.113810	1.14381	1.008413	0.971746	1.005238	1.001429	1.004127	1.016984	1.048889	77.209841	683.320635
std	0.193477	0.073431	0.202563	0.161639	0.17942	0.120593	0.080973	0.093375	0.051988	0.130935	0.085735	0.075586	168.509374	1821.582348
min	0.750000	0.940000	0.700000	1.000000	1.00000	0.870000	0.870000	0.900000	0.950000	0.820000	0.830000	1.000000	1.980000	5.900000
25%	0.880000	0.940000	1.000000	1.000000	1.00000	0.870000	0.870000	0.900000	0.950000	0.910000	1.000000	1.000000	8.650000	40.500000
50%	1.000000	1.000000	1.070000	1.060000	1.06000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	25.000000	98.000000
75%	1.150000	1.040000	1.300000	1.110000	1.21000	1.150000	1.000000	1.100000	1.000000	1.100000	1.100000	1.080000	60.000000	438.000000
max	1.400000	1.160000	1.650000	1.660000	1.56000	1.300000	1.150000	1.210000	1.140000	1.240000	1.240000	1.230000	1150.000000	11400.000000

Fig 3.3 COCOMO dataset description

There are eighteen independent and one dependent feature i.e., ‘actual’. We have applied feature selection technique and the features which are less correlated to the effort are dropped from the dataset.

3.1.4 Desharnais Dataset

Desharnais dataset was extracted from PROMISE repository. It is composed of 81 software projects with 12 features



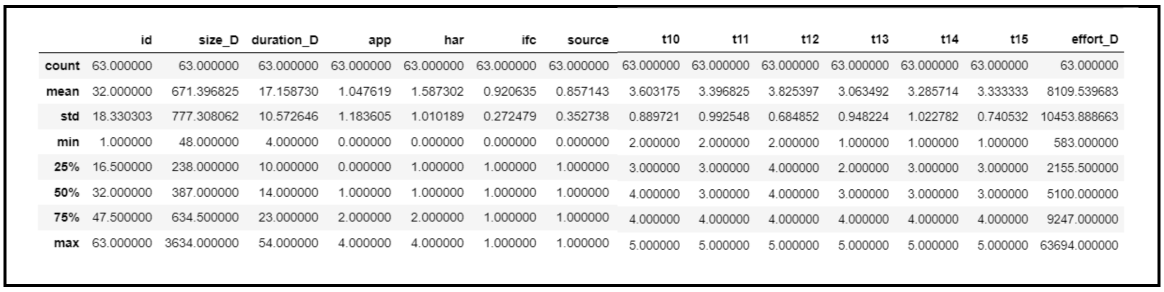
	id	Project	TeamExp	ManagerExp	YearEnd	Length	Effort	Transactions	Entities	PointsNonAdjust	Adjustment	PointsA
count	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.00
mean	41.000000	41.000000	2.185185	2.530864	85.740741	11.666667	5046.308642	182.123457	122.333333	304.456790	27.629630	289.23
std	23.526581	23.526581	1.415195	1.643825	1.222475	7.424621	4418.767228	144.035098	84.882124	180.210159	10.591795	185.76
min	1.000000	1.000000	-1.000000	-1.000000	82.000000	1.000000	546.000000	9.000000	7.000000	73.000000	5.000000	62.00
25%	21.000000	21.000000	1.000000	1.000000	85.000000	6.000000	2352.000000	88.000000	57.000000	176.000000	20.000000	152.00
50%	41.000000	41.000000	2.000000	3.000000	86.000000	10.000000	3647.000000	140.000000	99.000000	266.000000	28.000000	255.00
75%	61.000000	61.000000	4.000000	4.000000	87.000000	14.000000	5922.000000	224.000000	169.000000	384.000000	35.000000	351.00
max	81.000000	81.000000	4.000000	7.000000	88.000000	39.000000	23940.000000	886.000000	387.000000	1127.000000	52.000000	1116.00

Fig 3.4 Desharnais dataset description

There are eleven independent and one dependent feature i.e., ‘Effort’. We have applied feature selection technique and the features which are less correlated to the effort are dropped from the dataset. The features which we selected for our research are ‘Length’, ‘Transactions’, ‘Entities’, ‘PointsNonAdjust’, ‘PointsAjust’.

3.1.5 Maxwell Dataset

The Maxwell dataset comprises information on a number of projects involving Finnish trade banks. The Maxwell dataset contains 63 projects data with 25 different attributes.



	id	size_D	duration_D	app	har	ifc	source	t10	t11	t12	t13	t14	t15	effort_D
count	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000	63.000000
mean	32.000000	671.396825	17.158730	1.047619	1.587302	0.920635	0.857143	3.603175	3.396825	3.825397	3.063492	3.285714	3.333333	8109.539683
std	18.330303	777.308062	10.572646	1.183605	1.010189	0.272479	0.352738	0.889721	0.992548	0.684852	0.948224	1.022782	0.740532	10453.888663
min	1.000000	48.000000	4.000000	0.000000	0.000000	0.000000	0.000000	2.000000	2.000000	2.000000	1.000000	1.000000	1.000000	583.000000
25%	16.500000	238.000000	10.000000	0.000000	1.000000	1.000000	1.000000	3.000000	3.000000	4.000000	2.000000	3.000000	3.000000	2155.500000
50%	32.000000	387.000000	14.000000	1.000000	1.000000	1.000000	1.000000	4.000000	3.000000	4.000000	3.000000	3.000000	3.000000	5100.000000
75%	47.500000	634.500000	23.000000	2.000000	2.000000	1.000000	1.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	9247.000000
max	63.000000	3634.000000	54.000000	4.000000	4.000000	1.000000	1.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	63694.000000

Fig 3.5 Maxwell dataset description

There are 24 independent features and one dependent feature i.e., ‘effort_D’. We have applied feature selection technique and the features which are less correlated to the effort are dropped from the dataset. There are 14 features which we selected for our research.

3.2 Ensemble Models

Ensemble techniques integrate models provided by different models to produce superior outcomes. When compared to a single model, ensemble techniques often produce more accurate findings [33]. The two commonly used ensemble models are bagging and AdaBoost. In our research we have used these two ensemble models and tune the ‘base_learner’ hyperparameter with three machine learning models: elastic net, random forest and SVR.

3.2.1 Bagging

Bootstrap aggregation is sometimes known as 'bagging.' It's a meta-algorithm that improves the accuracy of machine learning models for classification and regression. It aids to prevent overfitting by reducing variance. It's most typically seen in decision tree methods. A subdivision of the model averaging method is bagging [34].

3.2.2 AdaBoost

Boosting is an ensemble technique used to build a robust classifier from a huge number of weak ones. It's done by joining together weak models to generate a model. To begin, a model is built using the training data. The second model is then created in order to resolve problems of the first. This approach is repeated unless the entire training data set has been successfully forecasted [34].

3.3 Machine Learning Models

Machine learning is a branch of computer science that allows machines to learn without being explicitly programmed. One of the most amazing technologies one has ever encountered is machine learning. As the name suggests, it allows the computer to understand, making it more human-like. Machine learning is presently used in far more locations than one might imagine.

3.3.1 Decision Tree Regressor

The Decision Tree is one of the most popular and useful supervised learning models. It has the ability to handle both regression and classification problems. Decision tree regression analyses an object's attributes and trains a model in the shape of a tree to forecast future data and deliver meaningful continuous output [35].

3.3.2 Random Forest

Random Forest is built on ensemble learning, which is an approach for addressing complicated problems by combining numerous classifiers and enhancing the performance of the model. Random Forest is a learner that averages the outcomes of numerous decision trees on different subsets of a dataset to enhance the dataset's anticipated accuracy. Instead, then relying on a decision tree, it collects forecasts from every tree and predicts the final output based on the majority of votes. It becomes more accurate as the forest grows denser [36].

3.3.3 Linear Regression

Linear regression is one of a supervised machine learning algorithm. It's largely utilised for forecasting and figuring out how features interact. Different regression models look at various types of correlations between dependent and independent features and employ various amounts of independent features Given an independent feature (x), linear regression is used to predict the value of a dependent feature (y). A linear relationship between x (input) and y (output) is discovered using this regression technique. [37].

3.3.4 Lasso Regression

This is a regularisation strategy that uses a Shrinkage method, also known as the penalised regression method, to choose features. The Least Absolute Shrinkage and Selection Operator, or Lasso, is a regularisation and model selection tool. Lasso regression refers to a model that employs the L1 regularisation technique [38].

3.3.5 Ridge Regression

Ridge regression, like lasso regression, constrains the coefficients by inserting a penalty component. Unlike lasso regression, ridge regression uses the square of the coefficients. L2 Regularization is another term for ridge regression [38].

3.3.6 Elastic Net

To regularise regression models, elastic net linear regression employs both lasso and ridge penalties. The methodology combines the lasso and ridge regression methods to improve the regularisation of statistical models by learning from their flaws. Variable selection and regularisation are done simultaneously with the elastic net approach [39].

3.3.7 SVR

SVR is a machine learning regression technique that relies on SVMs. Support vector techniques to machine learning tasks have a huge variety of applications due to their performance as well as the fact that they can be effectively linked with "kernel functions," allowing for a true nonlinear transformation of the input data space. Within a given threshold value, SVR tries to match the best line (distance between hyperplane and boundary line).

3.4 SMOTER

Many learning algorithms have issues with imbalanced domains. These issues are distinguished by an uneven proportion of cases accessible for the most critical ranges of the target variable to the user. To combat this problem, SMOTE is a well-known algorithm for classification problems. The basic idea behind this method is to leverage the cases' closest neighbours to artificially insert additional minority class examples. Furthermore, the majority of class examples are under-represented, resulting in a more balanced data collection. Torgo [40] proposed a new algorithm SMOTER which is a variant of the SMOTE algorithm that addresses the problem of imbalanced domains in regression applications [41].

3.5 Performance Metrics

The next step is to verify the model's effectiveness on test datasets after performing feature selection, applying different models, and obtaining the output in the form of a class or a continuous value. Machine learning models are evaluated using a variety of performance criteria. We used three performance indicators in our research: MMRE, MdMRE, and PRED (25).

3.5.1 MRE

MRE is a well-known criterion for picking the best software effort prediction model from a pool of competing models. The difference between actual and planned effort is calculated by MRE. It is used to find if differences in performance are statistically significant [42].

$$MRE_i = \frac{|x_i - \hat{x}_i|}{x_i} \tag{5.1}$$

(Where x_i , $\hat{x}_i$ are the actual and predicted values, respectively, for test instance i).

3.5.2 MMRE

An estimate of the effort is the mean of this distribution, that gives the relative and mean relative errors. It has become the de facto standard for evaluating the precision of software prediction models. The model with the lowest MMRE is the one to be used [43].

$$MMRE = \frac{\sum_{i=1}^{N} MRE_i}{N} \tag{5.2}$$

3.5.3 MdMRE

MdMRE value provides the median of the relative error.

$$MdMRE = median\left(MRE_{1,} MRE_2, MRE_3, \dots\right) \tag{5.3}$$

3.5.4 PRED(25)

PRED(25), is defined as the percentage of forecasts that are within 25% of the actual values, is a popular alternative error metric [42]. This phrase is commonly used in the literature to refer to the percentage of projects that are accurate to a particular degree.

$$PRED(25) = \frac{100}{N}\sum_{i=1}^{N}\begin{cases} 1, & if\ MRE_i \leq \frac{25}{100} \\ 0, & otherwise. \end{cases} \quad (5.4)$$

CHAPTER 4

PROPOSED WORK

The purpose of our first research is to compare the usage of two common ensemble approaches (bagging and boosting) to reduce the error for prediction. We used random forest, SVR and elastic net as the base learners. In this study Albrecht, Desharnais, and China datasets were used for experimentation. We also performed feature selection and considered only those features that have strong correlation with target feature i.e., effort. The MMRE and PRED(25) results demonstrate that utilising elastic net as the base learner for AdaBoost outperforms the other models.

Another research was carried out to study the impact of SMOTER to predict effort estimation by using machine learning algorithms. The different machine learning models that we have applied are decision tree regressor, random forest, linear regression, lasso regression and ridge regression. In this study we used three datasets i.e., China, Maxwell and COCOMO81. The two-performance metrics MMRE and PRED(25) were used to evaluate the results of each model using SMOTER and without using SMOTER. We have seen a significant decrease in error of each model after applying SMOTER.

4.1 Flow Diagram of Comparative Analysis of Ensemble Models for Software Effort Estimation.

For our study we selected three freely available datasets namely Albrecht dataset, Desharnais dataset and China dataset. The framework of research methodology is represented in Fig 4.1. Firstly, we imported all the necessary libraries and these three datasets. After importing the datasets, the pre-processing technique was applied for removing null values and negative values if exist.

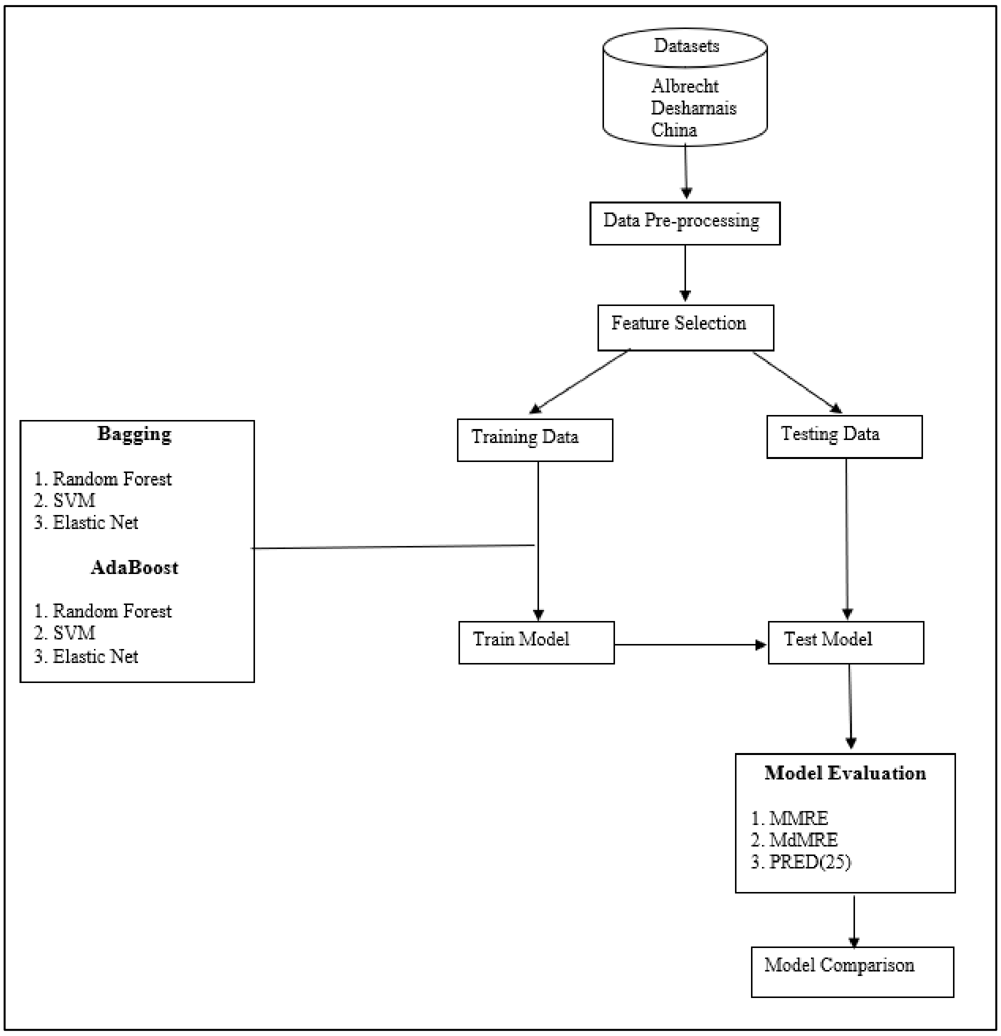


Fig 4.1 Framework of Comparative Analysis of Ensemble Models for Software Effort Estimation.

4.1.1 Feature Selection

After importing necessary data next step is feature selection.

To train a model, we collect massive amounts of data to assist the machine in learning better. Typically, a huge amount of the data obtained is noise, and some of the features in our dataset may not have a substantial impact on our model's performance. Furthermore, having a large amount of data can slow down the training process, resulting in a slower model. This useless data may also cause the model to learn incorrectly. Apart from selecting the appropriate model for our data, we must also select the appropriate data for our model.

Feature selection is a technique for limiting the input features to your model by removing noise and only considering useful characteristics. It's the process of selecting suitable machine learning model based on the type of problem you're trying to solve. This is achived by adding or eliminating crucial traits without changing their meaning. It improves in the reduction of data noise as well as the size of input data [44].

Feature selection was done with the help of correlation matrix. All the independent features having correlation greater than 0.5 with our dependent feature ‘effort’ considered for experimentation and rest of the features are dropped.

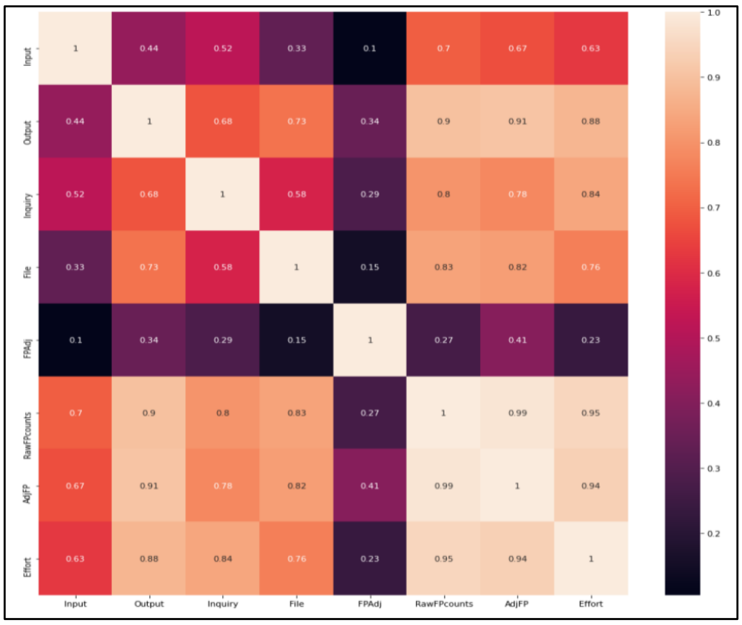


Fig 4.2 Correlation matrix of Albrecht datset

Fig 4.2, is the correlation matrix of Albrecht dataset and the features having correlation greater than 0.5 with target variable are considered. We have selected six independent features for our research.

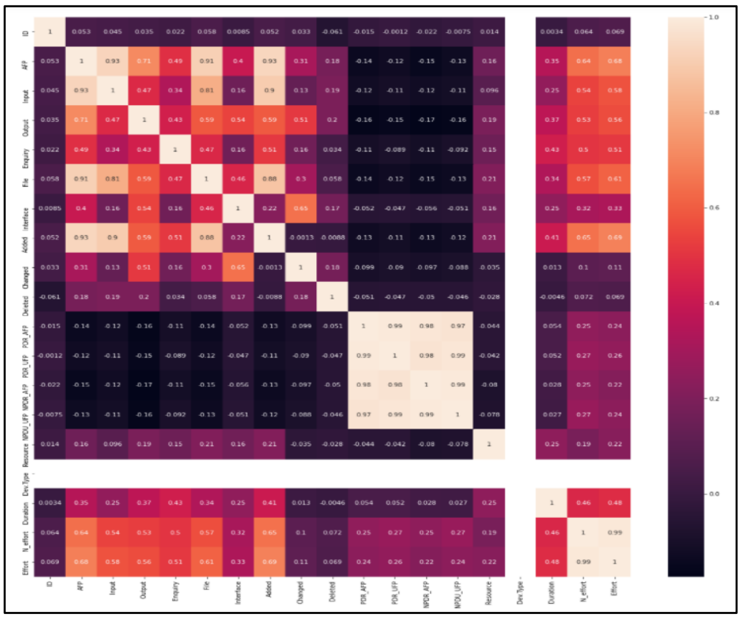


Fig 4.3 Correlation matrix of China datset

Fig.4.3, is the correlation matrix of China dataset and seven independent features which has a strong correlation i.e., correlation value greater than 0.5 with dependent feature ‘effort’ are selected.

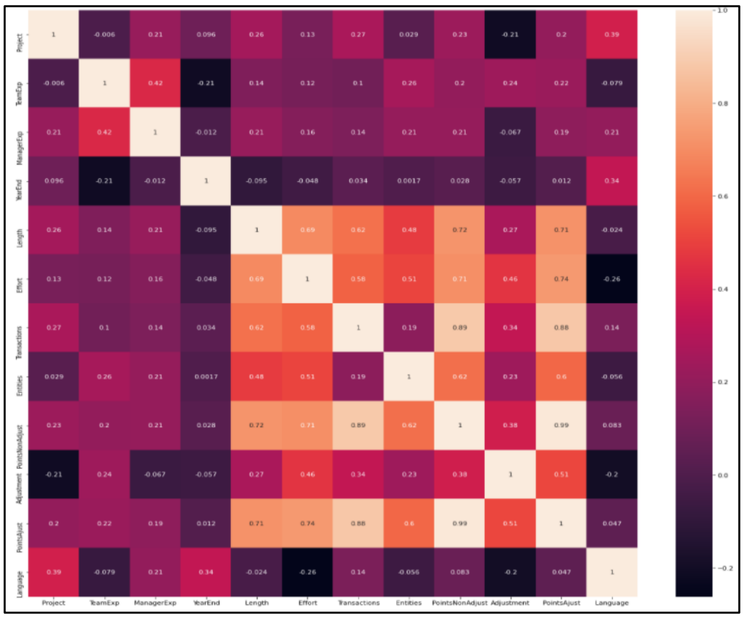


Fig 4.4 Correlation matrix of Desharnais datset

Fig 4.4, shows the correlation matrix of Desharnais dataset and five features were selected having correlation greater than 0.5 with effort.

4.1.2 Train-Test Split

The train-test split is a technique for assessing a machine learning model’s performance. It can be used for both classification and regression tasks. A dataset is split into two subsets as part of the method. The training dataset is the first subset employ to fit the model. Instead, the model receives the dataset's input element, which generates predictions and we compares that prediction with the predicted values.

For our research we have split our dataset in 70-30 ratio. 70% of the dataset was considered for training our models and 30% of the dataset considered for testing our result.

4.1.3 Applying Ensemble Models with Hyperparameter Tuning

After splitting our data, we train our data with the ensemble models. For predicting the software effort, we have used two ensemble models i.e., bagging and AdaBoost. Firstly, we have applied randomized search CV with a cross validation of 10 on three machine learning models i.e., random forest, SVR and elastic net and obtained the best parameters from them. The parameters that gave us the most accurate values of effort was further considered and were passed as the base learners for bagging and AdaBoost. We have applied randomized search CV with cross validation value of 5 on bagging and AdaBoost. In this study we have compared the results on the basis of three performance measures i.e., MMRE, MdMRE and PRED(25).

4.2 Flow Diagram of Software Effort Estimation Using Synthetic Minority Over-Sampling Technique for Regression (SMOTER)

For our research we choose three freely available datasets i.e., China, COCOMO81 and Maxwell. The framework of research methodology is represented in Fig 4.5. We imported all the libraries and these three datasets that were required for our study. After this we pre-processed our datasets by checking null and missing values. There was no missing value present in our datasets.

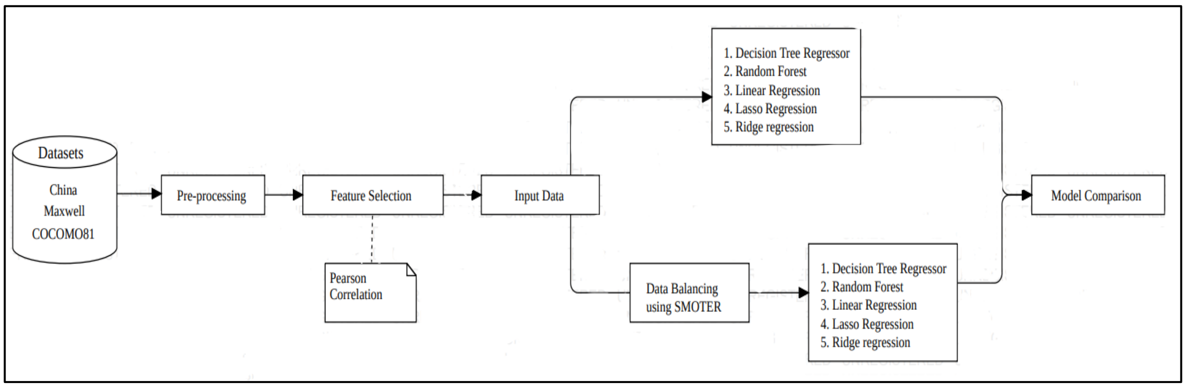
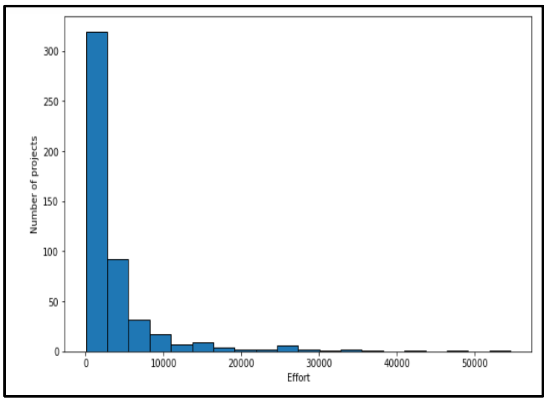


Fig 4.5 Framework of Software Effort Estimation Using Synthetic Minority Over-Sampling Technique for Regression (SMOTER)

4.2.1 Data Distribution



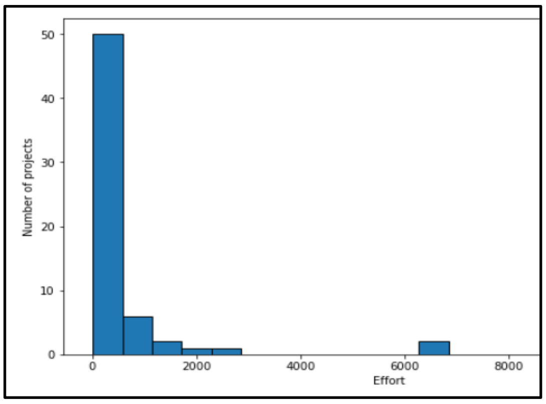


Fig 4.6 Data distribution of effort in China dataset

Fig 4.7 Data distribution of effort in COCOMO dataset

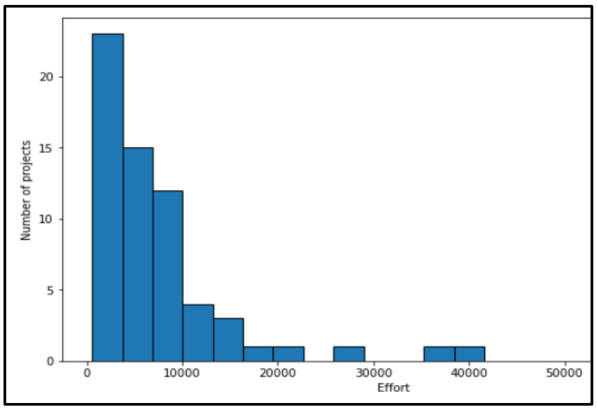


Fig 4.8 Data distribution of effort in Maxwell dataset

Data distribution of the target variable effort of China, COCOMO and Maxwell dataset is represented by Fig 4.6, Fig 4.7 and Fig 4.8 respectively. We can conclude from the histogram that the target variable is not evenly distributed. There are lot of missing data and distribution of data is not balance. To get better accuracy of our models the data should be balanced.

4.2.2 Feature Selection

Feature selection was done with the help of correlation matrix. All the independent features having correlation greater than 0.5 with our dependent feature ‘effort’ are considered for our further research.

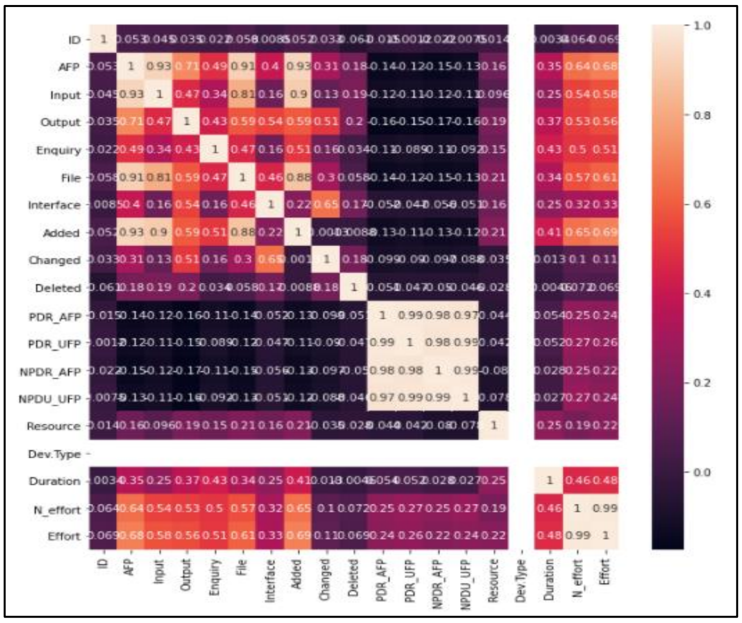


Fig 4.9 Correlation matrix of China dataset

Feature selection to remove the features that doesn’t have high impact on our target variable. Correlation matrix was used for the feature selection process. Correlation matrix of China dataset (Fig 4.9) shows that there are two features which have very low impact on the target variable so we removed those features from the dataset.

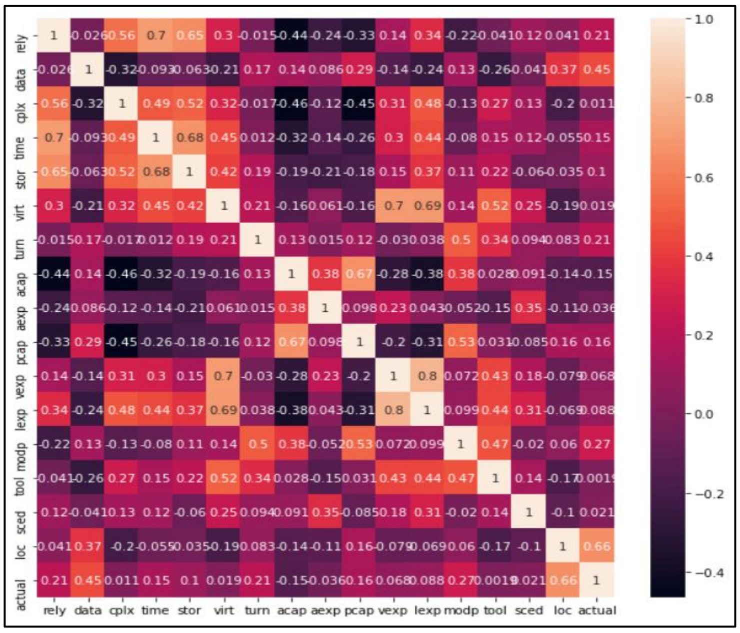


Fig 4.10 Correlation matrix of COCOCMO dataset

Fig 4.10 is a correlation matrix of COCOMO dataset. Correlation matrix reveals that there are seven features out of sixteen features which have low impact on target variable so, we can remove those features. So, for further study we considered only nine features.

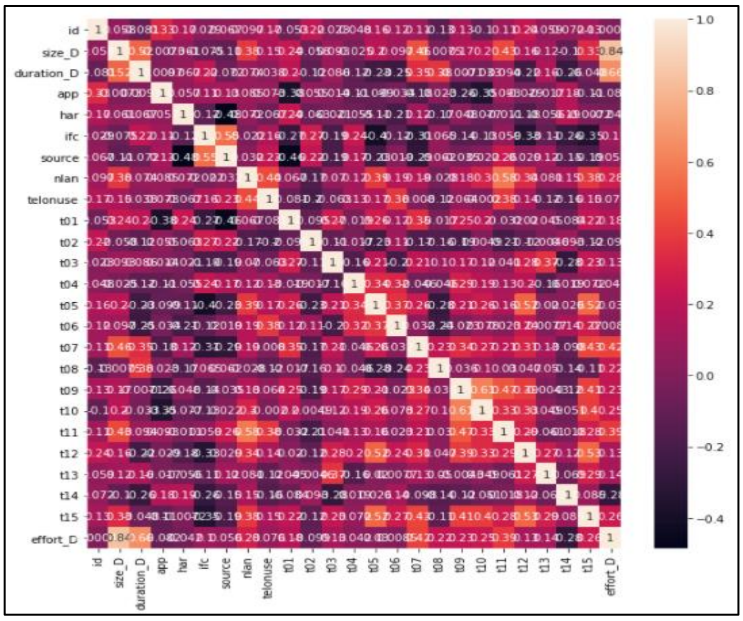


Fig 4.11 Correlation matrix of Maxwell dataset

Maxwell correlation matrix (Fig. 4.11) shows that we can drop nine features from the dataset as they have low correlation with target variable. We split data into training and testing set in 70:30 ratio respectively.

4.2.3 Applied Machine Learning Models

After selecting the necessary features and splitting our dataset in training and testing data we applied machine learning models to get the estimated value of the software effort. We have considered five machine learning algorithms in our study they are decision tree regressor, random forest, linear regression, lasso regression and ridge regression. After training our models with training dataset we test our models using testing. We used two performance metrics to evaluate the performance i.e., MMRE and PRED(25).

4.2.4 Applied Machine Learning Models after Using SMOTER

In order to improve the results of our models and to study the impact of data balancing on our models we applied SMOTER which is a data balancing technique.
After balancing our data with the help of SMOTER we again train and test our machine learning models on the balanced dataset. Later check the performance of our models and compared the results with the previous results.

CHAPTER 5

RESULT ANALYSIS

We have acquired the results of ensemble models with different base learners on the three selected datasets. For data balance, we used an over-sampling technique that resulted in a considerable reduction in error. For comparing the performance of our models we have used two performance metrics namely MMRE and PRED(25) values. The results are shown below with the help of tables and bar graphs.

5.1 Result of Comparative Analysis of Ensemble Models for Software Effort Estimation

The MMRE value was used to determine the difference between the estimated and actual effort. To get a better prediction, the difference between the two should be as minimum as possible. Table 5.1, shows the result of Albrecht dataset. AdaBoost with elastic net as the base learner gives us the best result for Albrecht dataset with the MMRE value of 0.373 which is the least among the rest of the models followed by bagging with elastic net i.e., 0.460. AdaBoost is a boosting technique which offers an iterative approach to turn weak classifiers into strong ones by learning from their mistakes. It improves prediction accuracy by learning from its mistakes and correcting them.

For Albrecht dataset AdaBoost with elastic net shows the best results.

China dataset results are shown in Table 5.2. AdaBoost with elastic net as a base learner offers the best result for our China dataset with the MMRE value of 0.193 which is the least among the rest of the models followed by bagging with elastic net as a base learner i.e., 0.209. AdaBoost with base learner as random forest and elastic net and bagging with base learner as random forest has more percentage of values close to the actual value i.e., 80%. The findings China dataset demonstrate that AdaBoost with base learner as elastic net predicts software effort with the maximum accuracy, as the relative error is quite low and the predicted values are very close to the actual value.

According to MMRE and PRED(25) value, AdaBoost with elastic net gives the best value for China dataset.

Table 5.3, shows the results of Desharnais dataset. AdaBoost with base learner as elastic net gives us the best result for our Desharnais dataset with the MMRE value of 0.626 which is the best compared to the other models followed by bagging with elastic net as a base learner i.e., 0.710.

AdaBoost with base learner as random forest and SVR and bagging with base learner as SVR has more percentage of values close to the actual value i.e., 48%

According to MMRE value AdaBoost with elastic net gives the best value for Desharnais dataset.

Table 5.1 Results of Albrecht dataset for ensemble models

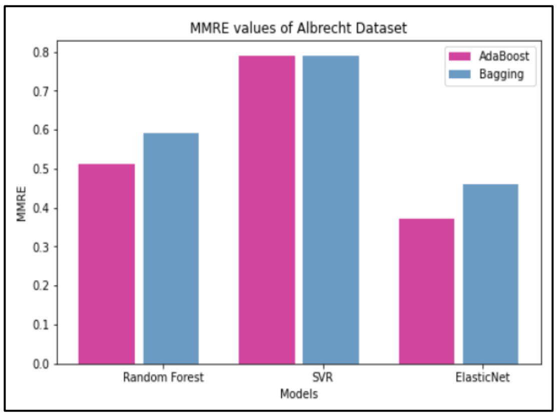
Models	MMRE	MdMRE	PRED (25)
RF	3.162	0.839	12.0
SVR	2.609	0.737	20.01
Elastic Net	1.655	0.522	20.0
AdaBoost with RF	0.513	0.423	25.0
AdaBoost with SVR	0.789	0.602	25.0
AdaBoost with Elastic Net	0.373	0.311	37.5
Bagging with RF	0.591	0.661	12.5
Bagging with SVR	0.790	0.593	25.0
Bagging with Elastic Net	0.460	0.428	25.0

Table 5.2 Results of China dataset for ensemble models

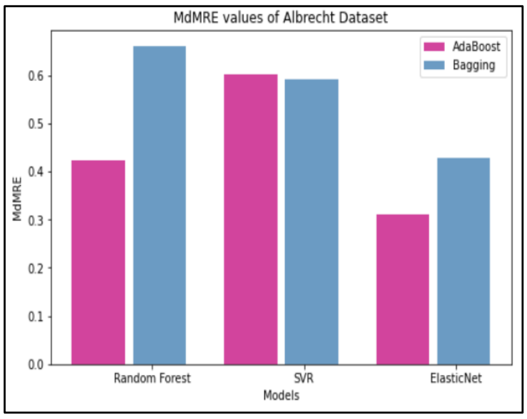
Models	MMRE	MdMRE	PRED (25)
RF	3.943	0.857	11.60
SVR	5.942	2.012	9.82
Elastic Net	3.433	0.968	12.0
AdaBoost with RF	0.221	0.093	80.66
AdaBoost with SVR	1.461	0.594	23.33
AdaBoost with Elastic Net	0.193	0.103	80.0
Bagging with RF	0.232	0.095	80.0
Bagging with SVR	1.163	0.598	29.33
Bagging with Elastic Net	0.209	0.110	76.66

Table 5.3 Results of Desharnais dataset for ensemble models

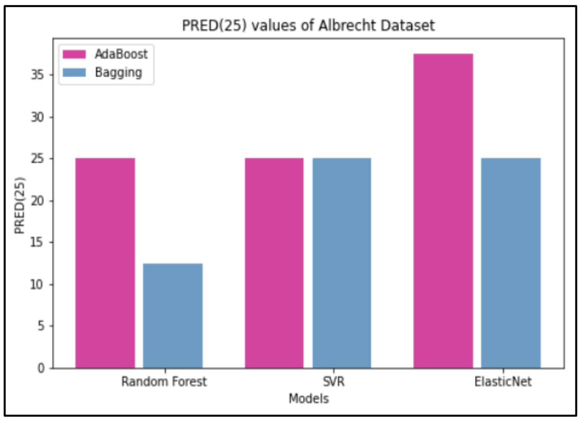
Models	MMRE	MdMRE	PRED (25)
RF	0.915	0.388	29.4
SVR	1.84	0.504	20.0
Elastic Net	0.990	0.993	0.0
AdaBoost with RF	0.764	0.264	48.0
AdaBoost with SVR	0.913	0.353	48.0
AdaBoost with Elastic Net	0.626	0.340	36.0
Bagging with RF	0.794	0.298	44.0
Bagging with SVR	0.934	0.353	48.0
Bagging with Elastic Net	0.710	0.381	36.0



(a)MMRE values



(b) MdMRE value



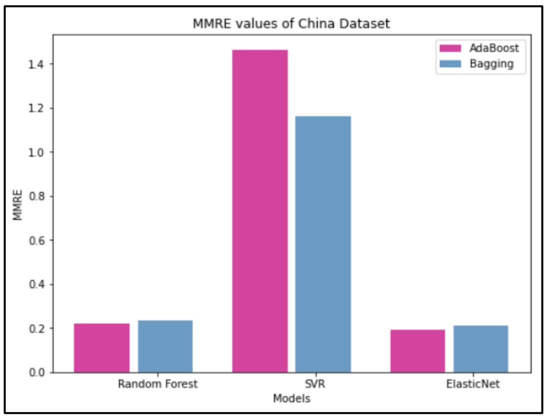
(c) PRED(25) value

Fig 5.1 Performance measures for Albrecht dataset

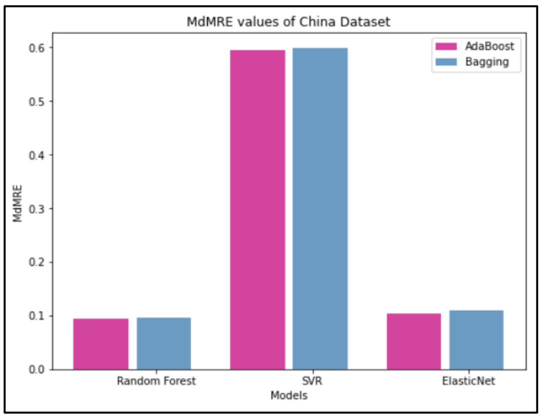
Fig 5.1(a), shows the graphical representation of MMRE value for Albrecht dataset. AdaBoost with random forest gives better MMRE value than boosting with random forest i.e., 0.513. SVR as the base learner for AdaBoost shows slightly better results as compared when used with bagging. AdaBoost with elastic net provides the best MMRE value for Albrecht dataset i.e., 0.373.

Fig 5.1(b) shows the result for MdMRE. Random forest and elastic net when used as base learner with AdaBoost give better MdMRE values than with bagging. AdaBoost with elastic net provides best MdMRE value i.e., 0.311.

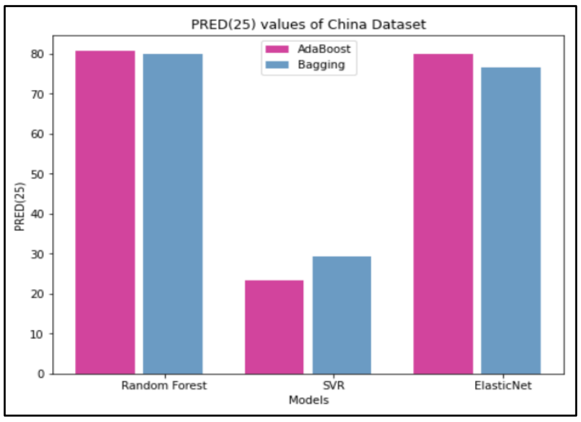
Fig 5.1(c) shows PRED(25) values. AdaBoost gives better results with all the base learners as compared to bagging. AdaBoost with elastic net provides the best result of PRED(25) i.e., 37.5.



(a)MMRE values



(b) MdMRE value



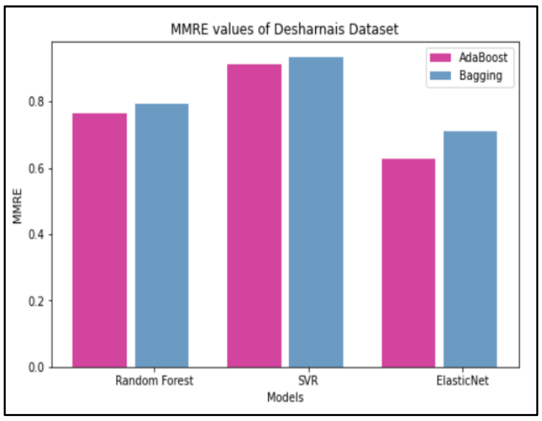
(c) PRED(25) value

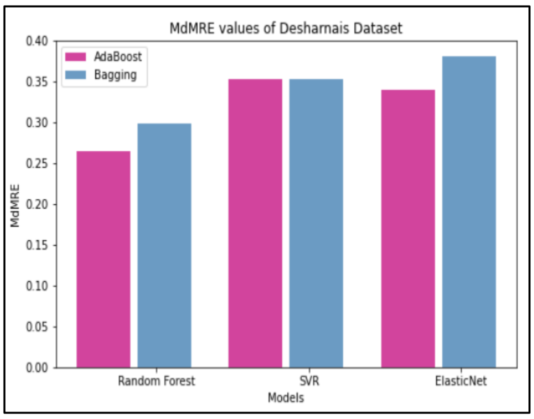
Fig 5.2 Performance measures for China dataset

Fig 5.2(a), shows the graphical representation of MMRE value for China dataset. AdaBoost with random forest gives slightly better MMRE value than boosting with random forest i.e., 0.221. SVR as the base learner for boosting gives better results as compared when used with AdaBoost. AdaBoost with elastic net provides the best MMRE value for China dataset i.e., 0.193.

Fig 5.2(b) shows the result for MdMRE. AdaBoost with all the base learners gives the better value as compared to bagging.

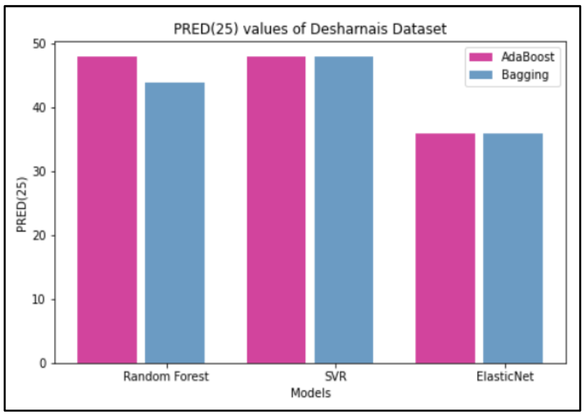
Fig 5.2(c) shows PRED(25) value. AdaBoost with random forest, AdaBoost with elastic net and bagging with random forest gives the best results for PRED(25) i.e., 80.





(a)MMRE values

(b) MdMRE value



(c) PRED(25) value

Fig 5.3 Performance measures for Desharnais dataset

Fig 5.3(a), shows the graphical representation of MMRE value for Desharnais dataset. AdaBoost with random forest gives better MMRE value than boosting with random forest i.e., 0.764. SVR as the base learner for AdaBoost gives slightly better results (0.913) as compared when used with bagging (0.934). AdaBoost with elastic net provides the best MMRE value for Desharnais dataset i.e., 0.626.

Fig 5.3(b) shows the result for MdMRE. Random forest and elastic net when used as base learner with AdaBoost give better MdMRE values than with bagging. SVR as the base learner provides the same value when used with bagging and AdaBoost i.e., 0.353.

Fig 5.3(c) shows PRED(25) values. AdaBoost with random forest, AdaBoost with SVR and bagging with SVR gives best PRED(25) value i.e., 48.

AdaBoost with elastic net provides the best results for all the datasets followed by bagging with elastic net.

5.2 Result of Software Effort Estimation Using Synthetic Minority Over-Sampling Technique for Regression (SMOTER)

Table 5.4 gives the result for China dataset, there is a significant decrease in MMRE value after applying SMOTER on our dataset. Random Forest gives the error of 0.09 before balancing our dataset, which was later decreased to 0.06 after using data oversampling technique (SMOTER). PRED(25) which is the predicted effort that falls in 25% of the actual value. This depicts the percentage of value that is close to 25% of our actual effort. All the models show an increase in PRED percentage after applying SMOTER. The highest percentage we got is 98 percentage for random forest.

Table 5.6 depicts the result for COCOMO81 dataset. Here all the models perform better after using oversampling technique for regression on the dataset. There is notable decrease in error i.e., MMRE value and an increase in number of results close to the actual value i.e., PRED(25) value.

Table 5.7 shows result for Maxwell dataset. There is a decrease in MMRE value for all the machine learning models and slight increase in PRED(25) value.

Table 5.4 Results of China dataset

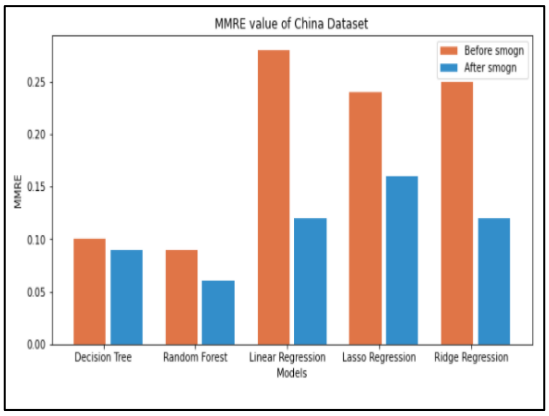
Models	**Before applying SMOTER**		**After applying SMOTER**	
	MMRE	***PRED(25)***	***MMRE***	***PRED(25)***
Decision Tree Regressor	0.10	94.0	0.09	94.5
Random Forest	0.09	98.6	0.06	98.6
Linear Regression	0.28	75.3	0.12	87.6
Lasso Regression	0.24	74.6	0.16	83.5
Ridge Regression	0.26	79.3	0.12	87.2

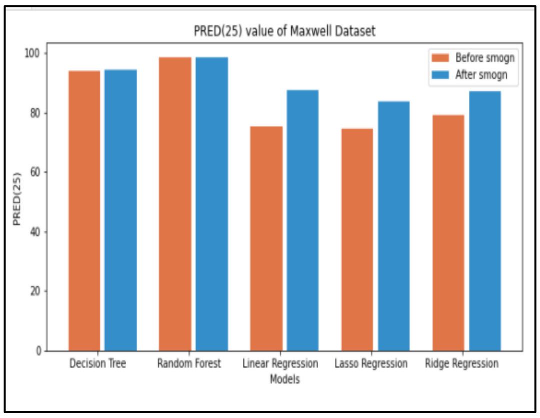
Table 5.5 Results of COCOMO dataset

Models	**Before applying SMOTER**		**After applying SMOTER**	
	MMRE	***PRED(25)***	***MMRE***	***PRED(25)***
Decision Tree Regressor	1.5	5.2	1.14	46.6
Random Forest	2.6	5.2	2.6	20.0
Linear Regression	15.1	10.5	4.8	20.0
Lasso Regression	14.4	15.7	4.5	20.0
Ridge Regression	7.9	10.5	2.3	13.3

Table 5.6 Results of Maxwell dataset

Models	**Before applying SMOTER**		**After applying SMOTER**	
	MMRE	***PRED(25)***	***MMRE***	***PRED(25)***
Decision Tree Regressor	0.76	31.5	0.60	56.6
Random Forest	1.31	31.5	0.49	66.6
Linear Regression	1.25	26.3	0.86	33.3
Lasso Regression	1.25	26.3	0.85	33.3
Ridge Regression	1.18	31.5	0.80	33.3

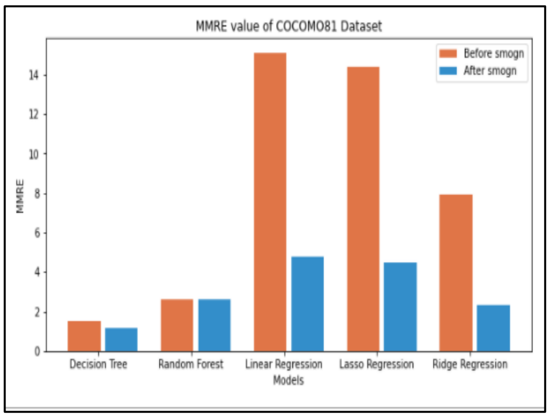


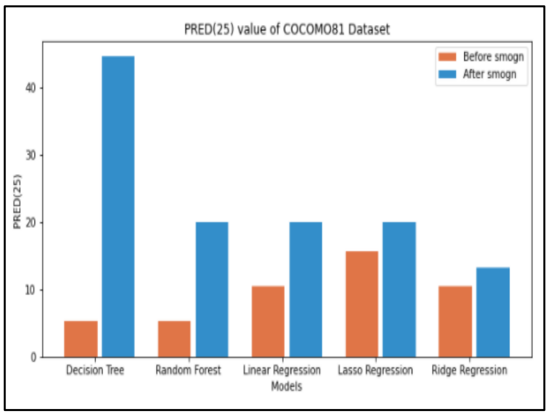


(a) MMRE values (b) PRED(25) values

Fig 5.4 Performance measures for China dataset.

Fig 5.4(a) depicts MMRE value and Fig 5.4(b) depicts the PRED(25) value. Both shows the better performance after using SMOTER

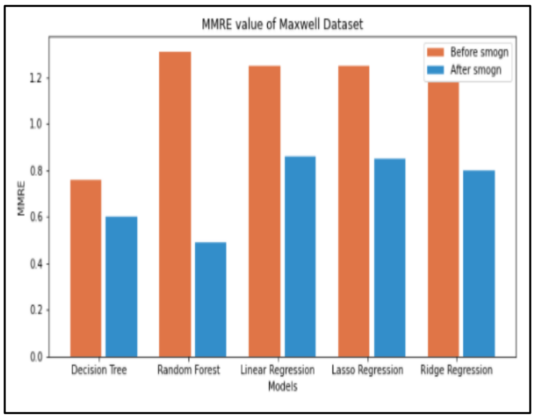


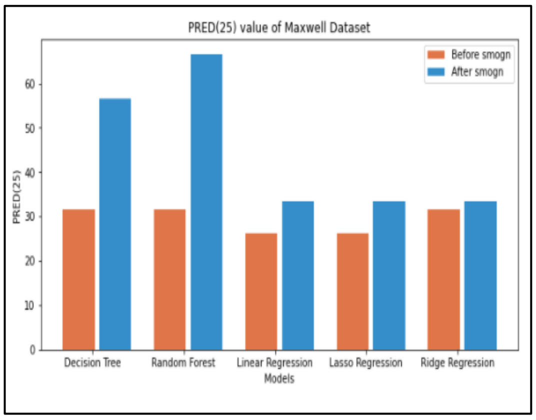


(a) MMRE values (b) PRED(25) values

Fig 5.5 Performance measures for COCOCMO dataset.

Fig 5.5 (a) depicts MMRE value and Fig 5.5(b) depicts the PRED(25) value. Linear regression and lasso regression shows a huge difference in MMRE value after applying SMOTER. Other models also perform better after balancing the data.





(a) MMRE values (b) PRED(25) values

Fig 5.6 Performance measures for Maxwell dataset.

Fig 5.6(a) depicts MMRE value and Fig 5.6(b) depicts the PRED(25) value. Both shows the better performance after using SMOTER but random forest gave the best result after data balancing.

The result shows that there is a significant decrease in error value for all the model after using oversampling technique.

5.3 Threats to Validity

Internal Validity – In this study we used correlation metrics as a feature selection strategy in this study, however depending on the type of dataset, other techniques might be used alternatively. Any parameter with a real number domain can have an endless number of values, however testing all conceivable parameter values in our study is unfeasible. Although more values can be examined, we believe that the values for each of the parameters studied are within a suitable range.
All models used in our research regarding the impact of SMOTER are generic machine learning models. We can hyper tune the parameters and apply randomized search CV or grid search CV to increase the accuracy. By doing hyperparameter tuning we can test various parameters of our models with different values and can find the best parameters which in turn can give us the better results.

External Validity – The external threat to the study's validity is whether the findings can be applied to other contexts. As a result, it's critical to determine how extensively the study's conclusions can be applied. For all datasets, AdaBoost with elastic net yields the best results, followed by bagging with elastic net. To address this threat, we selected three datasets including data from a variety of software projects. Because these datasets were acquired from many organisations, they are diverse and cover a wide range of features. Taking into account both category and numerical features could change the results of this study. Another threat is that we only utilised three machine learning models to tune the base learner hyperparameter; other models are recommended to generalise the results of this study.

Construct Validity – Measurement validity is one of the important threats under construct validity. In our research we used three performance metrics: MMRE, MdMRE and PRED(25). MMRE is one of the most widely used performance measure for numerical type of data.

CHAPTER 6

CONCLUSION

Estimating software effort accurately is very crucial step that has to be taken care during the planning phase of the software as failing to do so can affect the quality of the software. Machine learning and ensemble models have been employed in the past for predicting software effort. Hyperparameter tuning has also been observed in a number of researches. We extended these studies and tune the base learners of ensemble models. We have done our research on Albrecht, Desharnais and China dataset. Feature selection is a very important as it improves the accuracy and reduce the underfitting problem of the model. The features that were not correlated with our target variable were eliminated from the dataset. We used ensemble models and tune their base learners to predict the SEE. We have applied randomized search CV on our base learners i.e., random forest, SVR and elastic net to obtained the best parameters. Tuning the hyperparameters helped us to test the datasets with different values and get the best parameters. We passed those best parameters to our ensemble models, bagging and AdaBoost. The performance of models were calculated using MMRE, MdMRE, PRED(25).

The results show that the AdaBoost with elastic net surpasses rest of the models on all evaluation criteria especially for Albrecht and China dataset. As a result of these findings, we conclude that the AdaBoost with elastic net model is a promising approach for estimating software effort.

For our research on effort estimation using SMOTER, we have tried to decrease the error of effort prediction by improving the quality of the datasets. Research was done on China, Maxwell and COCOMO81 datasets by using machine learning models. In our study we implemented five models which are decision tree regressor, random forest, linear regression, lasso regression and ridge regression. In order to enhance the quality of datasets we tried to balance the dataset by using SMOTER. We calculated the MMRE and PRED(25) values of our models before and after applying SMOTER. We have seen that this over-sampling technique has reduced the error of our models. The result shows a remarkable decrease in error rate for each of the machine learning models after applying data balancing technique. All the datasets show the similar results.

So, we can conclude from our research that SMOTER can be used as one of the pre-processing techniques to balance our data for regression problems.

CHAPTER 7

FUTURE WORK

As for the further research we can work on AdaBoost with elastic net for other datasets also as it has given us the best results for all the three datasets that we have used. We can improve the research by tuning the hyperparameters of bagging and AdaBoost with other values. Feature selection techniques can also be improved by using different methods for selection of relevant features. We can try other weak learners as base learners in bagging and AdaBoost for SEE.

We can use ensemble machine learning models like bagging, boosting or stacking instead of machine learning models for effort prediction using SMOTER. In our study we have used only three freely available datasets but there are many paid software effort estimation datasets also available, we can extend our study on those datasets with ensemble models. As results of our research shows a significant decrease in error rate after applying SMOTER, so in order to get better results we can use this technique on other datasets having continuous and imbalanced type target feature.

REFERENCES

[1] G. Douzasa, F. Bacao and F. Last, “Improving Imbalanced Learning,” in Lecture Notes in Computer Science, 2013.

[2] P. Branco, L. Torgo and R. P. Ribeiro, “SMOGN: a Pre-processing Approach for Imbalanced Regression,” in 1st International Workshop on Learning with Imbalanced Domains - Theory and Applications, 2017.

[3] J. D. D. Santos, “Kite,” 21 08 2019. [Online]. Available: https://www.kite.com/blog/python/smote-python-imbalanced-learn-for-oversampling/. [Accessed 06 05 2022].

[4] Z. Abdelalia, H. Mustapha and N. Abdelwahed, “Investigating the use of random forest in software effort estimation,” in Intelligent Computing in Data Sciences, 2019.

[5] P. Pospieszny, B. Czarnacka-Chrobot and A.Kobyliński, “An effective approach for software project effort and duration,” The Journal of Systems & Software, vol. 137, pp. 184-196, 2017.

[6] P. Rijwani and S. Jain, “Enhanced Software Effort Estimation using Multi Layered FeedForward Artificial Neural Network Technique,” in Information Processing, 2016.

[7] S. G. MacDonell and M. J. Shepperd, “Combining techniques to optimize effort predictions,” The Journal of Systems and Software, vol. 66, no. 2, pp. 0164-1212, 2001.

[8] H. Ahmed, “Towards Data Science,” 07 07 2020. [Online]. Available: https://towardsdatascience.com/regression-for-imbalanced-data-with-application-edf93517247c. [Accessed 12 05 2022].

[9] A. Fernandez, S. Garcia, F. Herrera and N. V. Chawla, “SMOTE for Learning from Imbalanced Data: Progress andChallenges,” Journal of Artificial Intelligence Research, vol. 61, pp. 863-905, 2018.

[10] A. Mallik, S. Kumar and B. Panda, “Link prediction in complex networks using node centrality and light gradient boosting machine,” World Wide Web, 2022.

[11] R. Malhotra, “A Systematic Review of Machine Learning,” Applied Soft Computing, vol. 27, pp. 504-518, 2014.

[12] M. Kumar and K. S., “Predicting Customer Churn Using Artificial Neural Network,” Engineering Applications of Neural Networks, vol. 1000, 2019.

[13] A. Kumari, P. Varshini and K. Anitha, “Predictive analytics approaches for software effort,” Indian Journal of science and Technology, vol. 13, no. 21, pp. 2094-2103, 2020.

[14] M. Lefley and C. J. Burgess, “Can genetic programming improve software effort estimationn? a comparative evaluation,” Information and Software Technology, vol. 43, no. 14, 2001.

[15] B. L. Petronio , A. L. Oliveira and S. R. Meira, “Software effort estimation using machine learning techniques with robust confidence intervals,” in 19th IEEE International Conference on Tools with Artificial Intelligence, 2007.

[16] C. L. Martin, J. L. Pasquier, C. M. Yanez and A. G. Tornes, “Software development effort estimation using fuzzy logic: A case study,” in Sixth Mexican International Conference on Computer Science (ENC’05), 2005.

[17] M. O. Elish, “Improved estimation of software project effort using multiple additive regression trees,” Expert System. Application, vol. 36, no. 7, 2009.

[18] S. K. Venkatesan and P. R., “Hyperparameters tuning of ensemble model for software effort,” Journal of Ambient Intelligence and Humanized Computing, vol. 12, p. 6579–6589, 2020.

[19] S. Bibi and I. Stamelos, “Selecting the appropriate machine learning techniques for the prediction of software development costs,” in Artificial Intelligence Applications and Innovations: 3rd IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI), Athens, Greece, 2006.

[20] J. Gallego, D. Rodríguez and M. Sicilia, “Software project effort estimation based on multiple parametric models generated through data clustering,” Journal of Computer Science and Technology, vol. 22, no. 3, 2007.

[21] G. Finnie, G. Wittig and J.-M. Desharnais, “A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models,” Journal of System and software, vol. 39, no. 3, 1997.

[22] G. R. Wittig and F. G. E., “AI tools for software development effort estimation,” in Proceedings of the 1996 International Conference on Software Engineering: Education and Practice., Washington, DC, USA, 1996.

[23] N. Sund, T. M. Khoshgoftaar and N. Seliya , “An empirical study of predicting software faults with case-based reasoning,” Software Quality Journal, vol. 14, no. 2, 2006.

[24] P. C. Pendharkar, “Probabilistic estimation of software size and effort,” Expert System Application, vol. 37, no. 6, 2010.

[25] W. Hoffmann and L. Radlinski,, “On predicting software development effort using machine learning techniques and local data on predicting software development effort using machine learning techniques and local data,” 2017.

[26] K. Dejaeger, W. Verbeke, D. Martens and B. Baesens, “Data Mining Techniques for Software Effort,” IEEE Transactions on Software Engineering, vol. 38, no. 2, pp. 375-397, 2012.

[27] K. Dejaeger, W. Verbeke, D. Martens and B. Baesens, “Data Mining Techniques for Software Effort,” IEEE Transaction on Software Engineering, vol. 38, no. 2, pp. 375-397.

[28] Y. L. a. J. W. Keung, “zenodo,” 2010. [Online]. Available: https://doi.org/10.5281/zenodo.268467. [Accessed 02 2022].

[29] F. H. Yun, “zenodo,” 2010. [Online]. Available: https://doi.org/10.5281/zenodo.268446. [Accessed 02 2022].

[30] B. Boehm, “promise,” 1981. [Online]. Available: http://promise.site.uottawa.ca/SERepository/datasets/cocomo81.arff. [Accessed 02 2022].

[31] M. Shepperd, “promise,” 2005. [Online]. Available: M. Shepperd, http://promise.site.uottawa.ca/SERepository/datasets/desharnais.arff. [Accessed 02 2022].

[32] Y. Li., “Zenodo,” 2019. [Online]. Available: https://doi.org/10.5281/zenodo.268461. [Accessed 02 2022].

[33] N. Demir, “Developers,” [Online]. Available: https://www.toptal.com/machine-learning/ensemble-methods-machine-learning.

[34] Soumya, “GeeksforGeeks,” 07 07 2021. [Online]. Available: https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/. [Accessed 2022 05 15].

[35] A. Das, “GeeksforGeeks,” [Online]. Available: https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/. [Accessed 15 05 2022].

[36] “java T point,” [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm. [Accessed 15 05 2022].

[37] M. Gupta, “GeeksforGeeks,” [Online]. Available: https://www.geeksforgeeks.org/ml-linear-regression/.

[38] “data camp,” 25 03 2022. [Online]. Available: https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression. [Accessed 15 05 2022].

[39] “Corporate Finance Institute,” [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/other/elastic-net/. [Accessed 15 05 2022].

[40] L. orgo, R. P. Ribeiro, B. Pfahringer and P. Branco, “SMOTE for Regression,” Artificial Intelligence, pp. 378-389.

[41] “RDocumentation,” [Online]. Available: https://www.rdocumentation.org/packages/UBL/versions/0.0.6/topics/SmoteRegress. [Accessed 15 05 2022].

[42] E. Kocaguneli, T. Menzies and J. W. Keung, “On the value of Ensemble Effort Estimation,” 2012.

[43] V. Anandhi and R. M. Chezian, “Regression Techniques in Software Effort Estimation Using COCOMO Dataset,” in International Conference on Intelligent Computing Applications, 2014.

[44] K. Menon, “Simpli learn,” [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/feature-selection-in-machine-learning.