# UROP Report

Submitted

By

**Masana Dhathri Praveenya-AP19110010217**

**Nandamuri Laasya Choudary-AP19110010248**

**Chebium Sai Pranay-AP19110010255**

**Mothukuri JayaBharath-AP19110010264**


**Department of Computer Science and Engineering**

**SRM University-AP, Andhra Pradesh, India**

**April - 2022**

# DATA SHEET

| | | |
|---|---|---|
| Roll Numbers | : | 1. AP191110010217<br>2. AP191110010248<br>3. AP191110010255<br>4. AP191110010264 |
| Name of the student | : | 1. Masana Dhathri Praveenya<br>2. Nandamuri Laasya Choudary<br>3. Chebium Sai Pranay<br>4. Mothukuri JayaBharath |
| Title of the Project | : | Software Maintainability Prediction Using Ensemble Learning Algorithms |
| Branch & Section | : | CSE-B |
| Batch | : | 2019-2023 |
| Start Date (MM/DD/YYYY) | : | 19-01-2022 |
| End Date (MM/DD/YYYY) | : | 29-04-2022 |
| Status of the project | : | Completed |
| Name of UROP mentor<br><br>(SRM Faculty) | : | Dr. Ramachandra Reddy B |

# ACKNOWLEDGEMENT

We would like to express our special thanks to our mentor “Dr. Ramachandra Reddy” for his patience, guidance and support in teaching the course and helping us to complete the project by clarifying all the doubts we got while completing it. The way our mentor taught us the topics are never forgettable and easy to learn in a short period of time. This project will be a big achievement which will improve our confidence and discipline for completing any work on time and using it in further conditions. We also thank each of our teammates for helping and encouraging each other to complete the project in a better way.

# TABLE OF CONTENTS

# I. Abstract

Software Maintenance is a long process and is the longest phase in the software development life cycle. Once the software is developed and delivered, maintenance plays an important role in the success of the software. The prediction of the effort required for software maintainability would result in effective management. In this paper, we used ensemble machine learning techniques to predict accurate effort required to maintain a software. Given two datasets, we implemented various machine learning algorithms to predict accuracy. The results show that the prediction using ensemble learning methods is more accurate due to less error. The least error in prediction is obtained while using GradientBoost Classifier. Therefore, a more accurate prediction is obtained by using GradientBoost machine learning algorithm.

# II. Introduction

A software project lifecycle consists of five different phases. They are planning, designing, development, testing, deployment and maintenance. The software project maintenance is the longest phase of a project life cycle and is the most costly phase as well. Maintenance of software requires a lot of effort. The prediction of software updation and maintainability can be useful to support and guide a few crucial factors like costs of different projects, decisions related to software and so on. As a result, we can have control over the future software maintenance.

Recent research studies based on software maintainability prediction have different approaches to the prediction. None of them have proven to be best under all conditions. There are quite a lot of datasets and performance of the model may vary from dataset to dataset. Number of methods are available which are used to predict the software maintainability. Few of them are : ensemble methods, regression methods and so on. Out of all these methods ensemble methods take advantage of the capabilities of their constituent computational intelligence models (base learners) toward a dataset to come up with more accurate or, at least, competitive prediction accuracy as compared to the individual models. They provide reliable predictions and are trustworthy. Therefore, there is a need for empirical evidence on the effectiveness of ensemble methods and the extent to which these ensembles enhance the accuracy, or in some cases deteriorate the prediction accuracy.

In this research, using ensemble methods and regression methods we predicted software maintainability. The objective of this research is to investigate and evaluate different ensemble methods and to compare them against individual models and also among themselves. For predicting the software maintainability we took use of two

famous datasets QUES and UIMS. By taking the metrics in these datasets we evaluated regression methods and ensemble methods. The metrics are Weighted Methods per Class (WMC), Depth of Inheritance (DIT), Number of Children (NOC), Lack of Cohesion in Methods (LCOM), Response For a Class (RFC), Message Passing Coupling (MPC), Data Abstraction Coupling (DAC), Number of Local Methods (NOM), and (SIZE1) traditional line of code, (SIZE2) total number of attributes and methods of a class. We divided our data into two parts. 80% of the data is used for training and 20% of the data is used for testing.

The rest of the paper is as follows. Section 2 reviews the related works. In Section 3 we provide an overview of algorithms used and proposed methods.In Section 4 definitions of ensemble methods are given. In Section 5 we provide all the performance measures used to evaluate the ensemble methods. Result analysis for every empirical study is given in Sect. 6. Result analysis consists of tables and bar charts. In Section 7 we present the conclusion. And at last we provide references.

## III. Related Works

There are various research works that have implemented different methods to predict software maintainability. Some studies have looked into the relationship between object oriented metrics and maintainability of object oriented software systems. Significant correlations between the both were found. These metrics can also be used to predict the software maintainability. (Al-Dallal 2013; Bandi et al. 2003; Briand et al. 2001; Fioravanti and Nesi 2001; Li and Henry 1993; Misra 2005).

Thwin and Quah (2005) conducted experiments which showed that general regression neural networks predict software maintainability better than Ward Network model. Koten and Gray (2006) concluded that naive bayes model can predict software maintainability better than regression based models for one system. Then, Elish and Elish (2009) further studied the work done by Zhou and Leung (2007) and evaluated that TreeNet method can yield better prediction accuracy over the already existing prediction models.
In recent times, ensemble methods have gained popularity and have shown promising results in improving the accuracy of prediction in single models (Braga et al. 2007; Sol-lich 1996). Ensemble methods have been used for various predictions in software projects. For example, software reliability prediction (Zheng 2009), software project effort estimation (Braga et al. 2007; Elish et al. 2013), and software fault prediction (Aljamaan and Elish 2009; Khoshgoftaar et al. 2003). They have also been used in many real life applications like face recognition (Gutta and Wechsler 1996; Huang et al. 2000),

OCR (Mao 1998), seismic signal classification (Shimshoni and Intrator 1998) and protein structural class prediction (Bittencourt et al. 2005).

To the best of our knowledge, ensemble methods have not been used for software maintenance prediction, except for the initial work reported in Aljamaan et al. (2013) and later significant extension of the same, published by Elish et al. (2015). This paper compares the accuracy of prediction of software maintainability using regular regression methods and ensemble learning methods (homogeneous and heterogeneous).

A single ensemble method is seen to give least error and more accuracy, which will be discussed as we proceed.

## IV. Proposed Method

In this research, we have considered regression algorithms and ensemble methods to evaluate and compare them among themselves and against individual algorithms. Regression methods we have taken into consideration are: linear regression, Support Vector Regression (SVR), Artificial Neural Network (ANN).

For ensemble methods we have taken Adaptive Boosting (AdaBoost), Gradient Boost, Extreme Gradient Boosting (XGBoost), CatBoost. By applying these algorithms on both of the datasets UIMS and QUES, we will be able to know the more accurate and precise model compared to previous models.

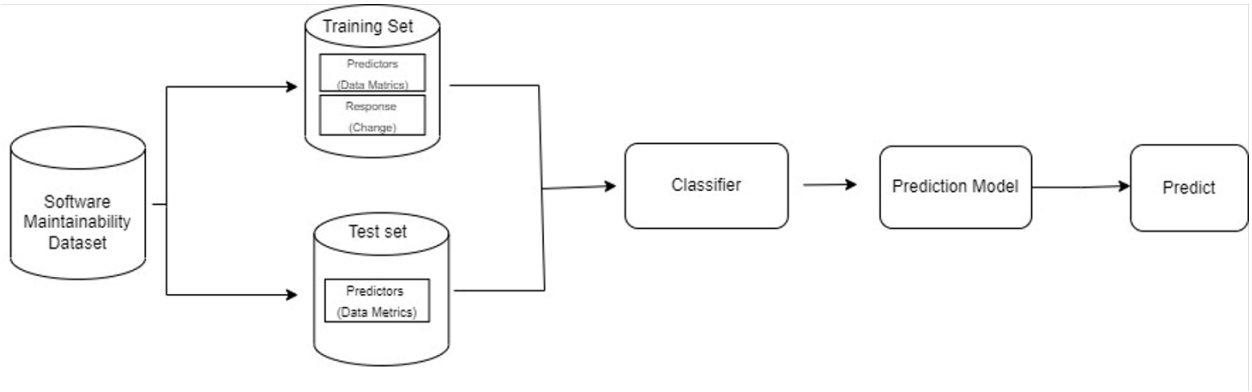


Fig 1: Proposed Method: Block Diagram

## V. The Ensemble Technique

An ensemble of computational intelligence models uses the outputs of all its individual constituent prediction models, each being assigned a certain priority level, and provides the final output with the help of an arbitrator[2][4]. There are single-model ensembles and multi-model ensembles. In single-model ensembles, the individual constituent prediction models are of the same type (for example, all of them could be radial basis function networks), but each with a randomly generated training set. Examples of single-model ensembles include Bagging and Boosting . In multimodel ensembles, there are different individual constituent prediction models[4].

The multi-model ensembles can be further classified, according to the design of the arbitrator, into linear ensembles and nonlinear ensembles[4]. In linear ensembles, the arbitrator combines the outputs of the constituent models in a linear fashion such as average, weighted average, etc. In nonlinear ensembles, no assumptions are made about the input that is given to the ensemble[4]. The output of the individual prediction models are fed into an arbitrator, which is a nonlinear prediction model such as a neural network which when trained, assigns the weights accordingly[4].

The proposed ensemble takes the advantage of the fact that individual prediction models have different errors across the used dataset partitions. The idea behind this ensemble is that across the dataset partitions, take the best model in training based upon a certain criterion in that partition. In this study, the criterion is mean magnitude of relative error (MMRE) [4].

**AdaBoost classifier:**

AdaBoost[6] is an ensemble method that trains and shows output in the form of trees in series. AdaBoost implements boosting, wherein a set of weak classifiers is connected in series such that each weak classifier tries to improve the classification of samples that were misclassified by the previous weak classifier[6]. In doing so, boosting combines weak classifiers in series to create a strong classifier. The decision trees used in boosting methods are called “stump” because each decision tree tends to be shallow models that do not overfit but can be biased[6]. An individual tree is trained to pay specific attention to the weakness of only the previous tree. The weight of a sample misclassified by the previous tree will be boosted so that the subsequent tree focuses on correctly classifying the previously misclassified sample[6]. AdaBoost is suited for imbalanced datasets but underperforms in the presence of noise. AdaBoost is slower to train.

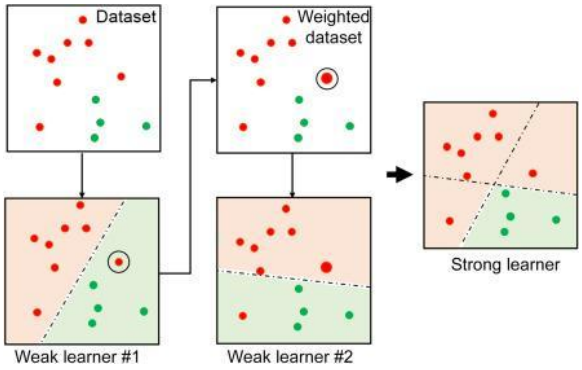

Fig2: AdaBoost Ensemble Method[6]

**Gradient Boost Regression:**

Gradient boosting[7] is a machine learning technique for regression and classification problems that produce a prediction model in the form of an ensemble of weak prediction models[7]. This technique builds a model in a stage-wise fashion and generalizes the model by allowing optimization of an arbitrary differentiable loss function. Gradient boosting basically combines weak learners into a single strong learner in an iterative fashion[7]. As each weak learner is added, a new model is fitted to provide a more accurate estimate of the response variable[7]. The new weak learners are maximally correlated with the negative gradient of the loss function, associated with the whole ensemble. The idea of gradient boosting is that you can combine a group of relatively weak prediction models to build a stronger prediction model.

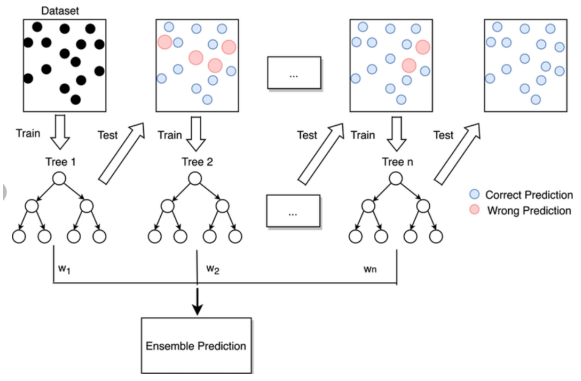

Fig3:Gradient Boosting approach[8]

**XGBoost Regression:**

XGBoost[9] is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.

In this algorithm, decision trees are created in sequential form . Weights play an important role in XGBoost[9]. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results[9]. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model[9]. It can work on regression, classification, ranking, and user-defined prediction problems.

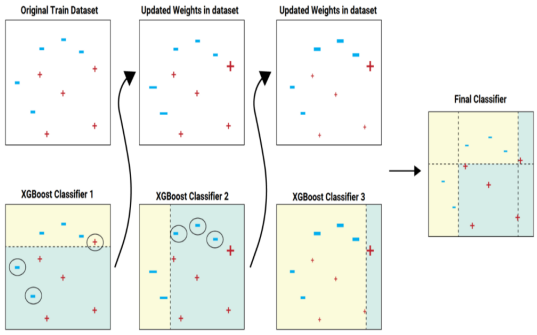


Fig 4: XGBoost method[10]

**CatBoost Regression:**

CatBoost[11] builds upon the theory of decision trees and gradient boosting. The main idea of boosting is to sequentially combine many weak models (a model performing slightly better than random chance) and thus through greedy search create a strong competitive predictive model[11]. Because gradient boosting fits the decision trees sequentially, the fitted trees will learn from the mistakes of former trees and hence reduce the errors[11]. This process of adding a new function to existing ones is continued until the selected loss function is no longer minimized[11].

In the growing procedure of the decision trees, CatBoost does not follow similar gradient boosting models. Instead, CatBoost grows oblivious trees, which means that the trees are grown by imposing the rule that all nodes at the same level, test the same predictor with the same condition, and hence an index of a leaf can be calculated with bitwise operations.

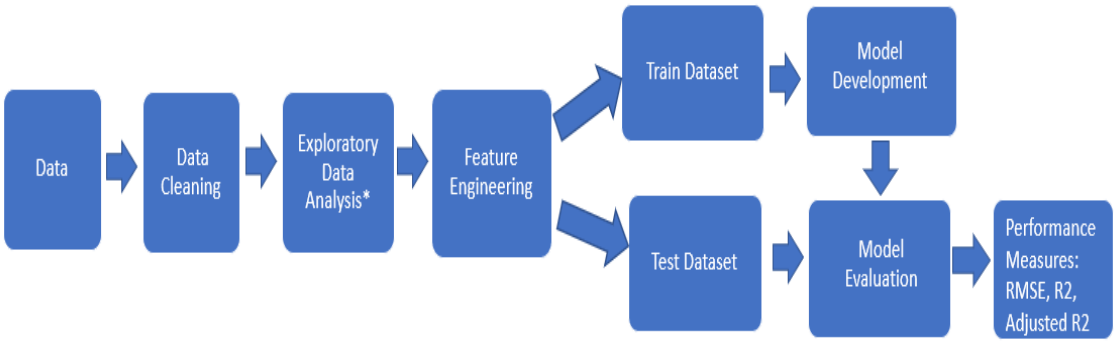


Fig 5: CatBoost approach[12]

## VI. Empirical Evaluation

This empirical evaluation aims to find out how much increase in software maintenance effort prediction accuracy a proposed ensemble method would offer over individual methods[2][4].

### 6.1 Datasets

We used two popular object-oriented software maintainability datasets published by Li and Henry : UIMS and QUES datasets.[2][4] These datasets are publicly available which makes our study verifiable, repeatable, and reputable. The UIMS dataset contains class-level metrics data collected from 39 classes of a user interface management system, whereas the QUES dataset contains the same metrics collected from 71 classes of a quality evaluation system.[2][4] Both systems were implemented in Ada. Both datasets consist of eleven class-level metrics: ten independent variables and one dependent variable.[2][4] The independent (input) variables are five Chidambar and Kemerer metrics : WMC, DIT, NOC, RFC, and LCOM; four Li and Henry metrics : MPC, DAC, NOM, SIZE2; and one traditional line of code metric. Table 1 provides a brief description for each metric.[2][4]

The dependent (output) variable is a maintenance effort proxy measure, which is the actual number of lines in the code that were changed per class during a 3-year maintenance period.[2][4] A line change could be an addition or a deletion. A change in the content of a line is counted as a deletion and an addition. Previous studies, on both datasets, indicate that both datasets have different characteristics, and therefore, considered heterogeneous and a separate maintenance effort prediction model is built for each dataset.[2][4]

| Metric | Description |
|---|---|
| WMC | Count of methods implemented within a class |
| DIT | Level for a class within its class hierarchy |
| NOC | Number of immediate subclasses of a class |
| RFC | Count of methods implemented within a class plus the number of methods accessible to an object class due to inheritance |
| LCOM | The average percentage of methods in a class using each data field in the class subtracted from 100% |
| MPC | The number of messages sent out from a class |
| DAC | The number of instances of another class declared within a class |
| NOM | The number of methods in a class |
| SIZE1 | The number of lines of code excluding comments |
| SIZE2 | The total count of the number of data attributes and the number of local methods in a class |

Fig6**.** Independent variables in the datasets[2][4]

### 6.2 Various Performance Measures

We used de facto standard and commonly used accuracy evaluation measures that are based on magnitude of relative error (MRE). These measures are mean magnitude of relative error (MMRE), standard deviation magnitude of relative error (StdMRE), and prediction at level q (Pred(q)). MMRE over a dataset of n observations is calculated as follows:[2][4]

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$$

where MREi is a normalized measure of the discrepancy between the actual value (xi) and the predicted value (xi) of observation i. It is calculated as follows:

$$MRE_i = \frac{|x_i - \widehat{x_i}|}{x_i}$$

Pred(q) is a measure of the percentage of observations whose MRE is less than or equal to q. It is calculated as follows:[2][4]

$$Pred(q) = \frac{k}{n}$$

where k is the number of observations whose MRE is less than or equal to a specified level q, and n is the total number of observations in the dataset. An acceptable value for level q is 0.3, as indicated in the literature. We therefore adopted that value.[2][4]

## VII. Result and Analysis

We used the 80:20 method of training and testing the model. We have taken two datasets QUES and UIMS into consideration and have predicted performance measures by taking the help of different algorithms.

Table 1 provides the results obtained from applying the individual algorithms on the QUES dataset. The individual algorithms we have taken in table 1 are linear regression, SVR, and ANN. We have calculated the performance measures MRE, RMSE, Pred(0.3). Out of all the models it can be observed that linear regression outperformed all the individual algorithms.

| | | MSE | RMSE | Pred(0.3) |
|---|---|---|---|---|
| QUES.csv | Linear Regression | 3770.42 | 61.40 | 0.06 |
| | SVR | 829.67 | 28.80 | 0.33 |
| | ANN | 2656.85 | 53.72 | 0.33 |

Table1: Individual Algorithms- QUES Dataset

Table 2 provides the results obtained from applying the ensemble methods for the QUES dataset. The ensemble methods we have used in this research are AdaBoost, Gradient Boost, XGBoost and CatBoost regression. We have calculated the performance measures MSE, RMSE, Pred(0.3) for all these methods. Out of all the methods, Gradient Boost outperformed compared to all the other ensemble methods.

| | | MSE | RMSE | Pred(0.3) |
|---|---|---|---|---|
| QUES1.csv | AdaBoost | 3357.96 | 56.57 | 0.0 |
| | Gradient Boost Regression | 1094.26 | 26.01 | 0.46 |
| | XGBoost Regression | 3455.05 | 58.62 | 0.06 |
| | CatBoost Regression | 2272.26 | 47.66 | 0.26 |

Table 2: Ensemble Methods- QUES Dataset

Fig. 4 shows a bar chart of the achieved Pred(0.30) value by each method. From the figure it can be seen clearly that the Gradient Boost regression achieved the best Pred(0.30).

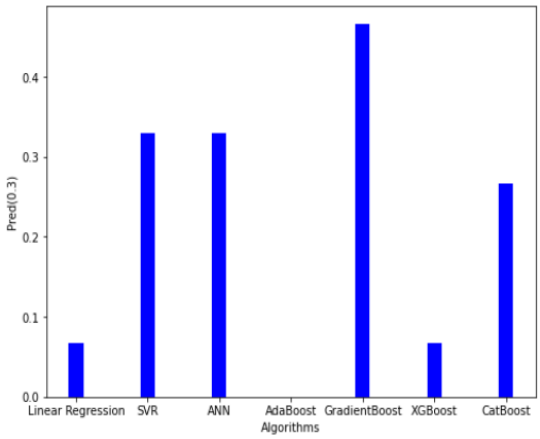


Fig 4: Pred(0.3) of QUES

Fig 5 shows a bar chart of the achieved RMSE values by each method for the QUES Data.

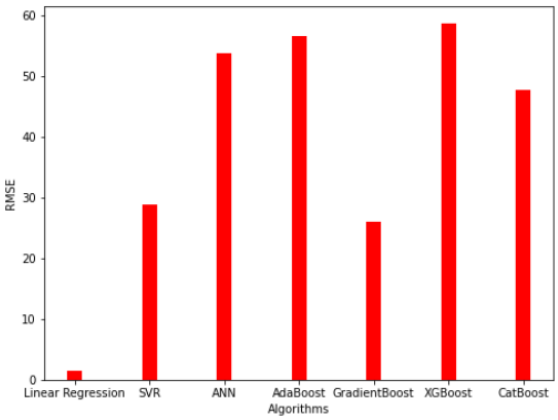


Fig 5: RMSE for Ques Dataset

Table 3 provides the results obtained from applying the individual algorithms on the UIMS dataset. The individual algorithms we have taken in table 2.3 are linear regression, SVR, and ANN. We have calculated the performance measures MRE, RMSE, Pred(0.3). Out of all the models it can be observed that linear regression outperformed all the individual algorithms.

| | | MSE | RMSE | Pred(0.3) |
|---|---|---|---|---|
| UIMS.csv | Linear Regression | 170.70 | 13.06 | 1.0 |
| | SVR | 6731.32 | 82.04 | 0.25 |
| | ANN | 8346.05 | 40.72 | 0.33 |

Table 3: Individual Algorithms- UIMS Dataset

Table 4 provides the results obtained from applying the ensemble methods for the UIMS dataset. The ensemble methods we have used in this research are AdaBoost, Gradient Boost, XGBoost and CatBoost regression. We have calculated the performance measures MSE, RMSE, Pred(0.3) for all these methods. Out of all the methods, Gradient Boost outperformed compared to all the other ensemble methods.

| | | MSE | RMSE | Pred(0.3) |
|---|---|---|---|---|
| UIMS.csv | AdaBoost | 514.52 | 22.90 | 0.5 |
| | Gradient Boost Regression | 975.0 | 30.57 | 0.5 |
| | XGBoost Regression | 138.44 | 21.22 | 0.62 |
| | CatBoost Regression | 524.25 | 22.89 | 0.62 |

Table 4:Ensemble Methods-UIMS Dataset

Fig. 6 shows a histogram of the achieved Pred(0.30) value by each model. From the figure it can be seen clearly that the Gradient Boost has the best Pred(0.30).

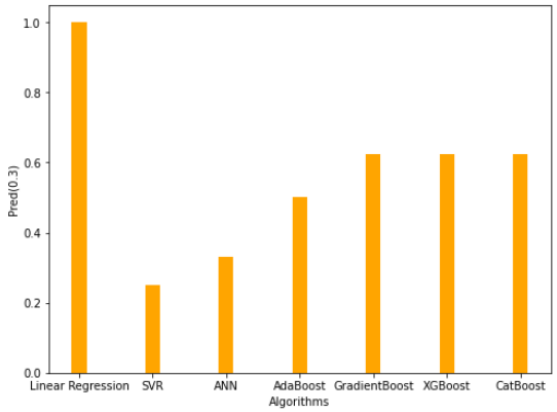


Fig 6: Pred(0.3) for UIMS Dataset

Fig. 7 shows a bar chart of the achieved Pred(0.30) value by each model.

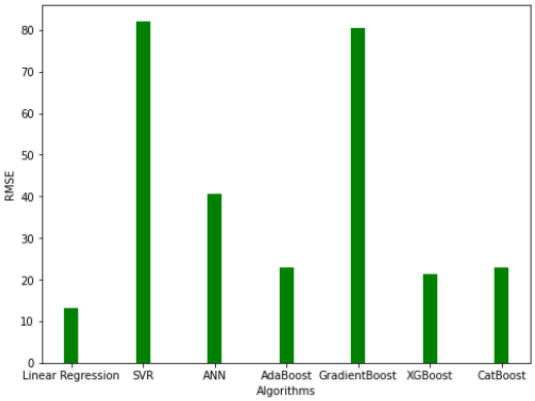


Fig 7: RMSE for UIMS Dataset

When considering the results from both datasets, there are a number of interesting observations. First, the results support that the performance of the individual prediction models may vary from dataset to dataset; the Gradient Boost regression was the best in the QUES and UIMS dataset. Finally, Gradient Boost generally achieved more accuracy compared to ensemble methods.

## VIII. Conclusion

In this research paper, we presented different ensemble methods and individual algorithms for predicting the software maintenance. For individual algorithms we have taken linear regression, SVR, and ANN to predict the performance measures and to compare them with the ensemble methods. In ensemble methods, four popular prediction models (AdaBoost, Gradient Boost, XGBoost, CatBoost) were taken into consideration. The prediction accuracy of the ensemble methods were investigated and evaluated using two publicly available software maintenance datasets (QUES and UIMS). After reviewing the results we got by evaluating the datasets using the given algorithms and methods, the results indicate that ensemble methods are providing more accurate predictions than individual algorithms thus it is more reliable. It is worth noting that ensemble methods are computationally more expensive than the individual algorithms but the benefits provided by the ensemble in terms of prediction accuracy.

After confirming that ensemble methods are more reliable than individual machine learning methods, we reviewed different ensemble methods separately to know the most reliable and best predicting method. After evaluating the methods, we discovered that Pred(0.3) and MMRE values are having better results for Gradient Boost compared to other ensemble methods (Adaboost, XGBoost, CatBoost). Therefore, Gradient Boost provides more accurate prediction of software maintenance out of all the ensemble methods.

## IX. References


1. Kaur, A., Kaur, K., & Malhotra, R. (2010). Soft computing approaches for prediction of software maintenance effort. *International Journal of Computer Applications*, *1*(16), 69-75.
2. Elish, M. O., Aljamaan, H., & Ahmad, I. (2015). Three empirical studies on predicting software maintainability using ensemble methods. *Soft Computing*, *19*(9), 2511-2524.
3. Riaz, M., Mendes, E., & Tempero, E. (2009, October). A systematic review of software maintainability prediction and metrics. In *2009 3rd international symposium on empirical software engineering and measurement* (pp. 367-377). IEEE.
4. Aljamaan, H., Elish, M. O., & Ahmad, I. (2013, June). An ensemble of computational intelligence models for software maintenance effort prediction. In

*International Work-Conference on Artificial Neural Networks* (pp. 592-603). Springer, Berlin, Heidelberg.
5. Baqais, A. A. B., Alshayeb, M., & Baig, Z. A. (2014). Hybrid intelligent model for software maintenance prediction.
6. Abirami, S., Kousalya, G., Balakrishnan and Karthick, R., 2022. Varied Expression Analysis of Children With ASD Using Multimodal Deep Learning Technique.
7. En.wikipedia.org. 2022. Gradient boosting - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Gradient_boosting> [Accessed 6 May 2022].
8. Zhang, Tao & Lin, Wuyin & Vogelmann, Andrew & Zhang, Minghua & Xie, Shaocheng & Qin, Yi & Golaz, Jean‐Christophe. (2021). Improving Convection Trigger Functions in Deep Convective Parameterization Schemes Using Machine Learning. Journal of Advances in Modeling Earth Systems. 13. 10.1029/2020MS002365.
9. Abirami, S., Kousalya, G., Balakrishnan and Karthick, R., 2022. *Varied Expression Analysis of Children With ASD Using Multimodal Deep Learning Technique*.
10. Blog.thinknewfound.com. 2022. [online] Available at: <https://blog.thinknewfound.com/2020/05/defensive-equity-with-machine-learning/xg-boost-final-01/> [Accessed 6 May 2022].
11. Medium. 2022. *CatBoost regression in 6 minutes*. [online] Available at: <https://towardsdatascience.com/catboost-regression-in-6-minutes-3487f3e5b329> [Accessed 6 May 2022].
12. Analytics Vidhya. 2022. *CatBoost | Predict Mental Fatigue Score using CatBoost*. [online] Available at: <https://www.analyticsvidhya.com/blog/2021/04/how-to-use-catboost-for-mental-fatigue-score-prediction/> [Accessed 6 May 2022].