MAJOR PROJECT REPORT

ON

"Software Maintainability Prediction using Machine Learning"

Submitted in Partial Fulfillment of Requirements for the Award of Degree of Bachelor of Technology in Computer Science and Engineering

Submitted to:

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI

Submitted by

Rahul Chaudhry

70716403216

Under the Guidance of

Dr. Anuradha Chug

UNIVERSITY SCHOOL OF INFORMATION, COMMUNICATION & TECHNOLOGY

Certificate of Originality

I hereby declare that the project work entitled **"Software Maintainability Prediction using Machine Learning"** submitted to the **GGSIPU, Delhi** is a record of an original work done by me under the guidance of **Dr Anuradha Chug (Assistant Professor) of USICT, GGSIPU, Delhi** and this project work is submitted in the partial fulfillment of the requirements for the degree of bachelors of technology in computer science engineering. The results embodied in this project have not been submitted to any other university or institute for the award of any degree or diploma.

Date:

Rahul Chaudhry

70716403216

I certify that the above statements made by the student are correct to the best of my knowledge and beliefs.

Date:

Dr Anuradha Chug

Assistant Professor

University School of Information, Communication and Technology

ACKNOWLEDGEMENT

The extensive endeavor and euphoria that accompany the successful completion of any project would be incomplete without the mention of the people who made it a success and whose constant guidance and encouragement crown all the effort. It was not only a technical endeavor but also the initialization of us into the world of industrial engineering field.

I express my heartfelt gratitude to **Dr Anuradha Chug** and Ms Shikha Gupta for their valuable guidance and cooperation throughout the whole project and I finally extent my gratitude to my friends and family whose constant support made me enthusiastic enough to accomplish the task and finally emerge with a successful project.

Rahul Chaudhry

70716403216

Table of Content

List of Figures

List of Tables

Abstract

Software Maintainability is one of the most significant aspects while assessing nature of the software product. It is characterized as the straightforwardness with which a software framework or segment can be changed to address issues, improve execution or different credits or adjust to a changed condition. Following the upkeep conduct of the software product is mind boggling. This is definitely the explanation that anticipating the expense and hazard related with upkeep after conveyance is very troublesome which is broadly recognized by the analysts and experts. While trying to address this issue quantitatively, the principle motivation behind this paper is to propose utilization of barely any machine learning calculations with a target to anticipate software maintainability and assess them. So as to contemplate and assess its presentation, two business datasets UIMS (User Interface Management Framework) and QUES (Quality Evaluation System) are utilized. The code for these two frameworks was written in Classical Ada. The UIMS contains 39 classes and QUES datasets contains 71 classes. To quantify the practicality, number of "CHANGE" is seen over a time of three years. We can characterize CHANGE as the quantity of lines of code which were included, erased or adjusted during a multiyear support that is all.

The objective of our investigation was to build appropriate model utilizing machine learning algorithms for the forecast of cost arranged to software maintainability which are anything but difficult to apply as well as could lessen the expectation mistakes to least.

The expectation execution of the machine learning calculations based models were surveyed and contrasted and winning models as far as R Squared values, MAE values and RMSE values. It was discovered that KNN and Ridge Regression model beat the predominant models as the least RMSE values was recorded. Most definitely, KNN (R- squared value = 0.64, RMSE value = 28.09) and Ridge regression (R- squared value = 0.64, RMSE value = 27.88) models are fundamentally better over others on the QUES dataset though GRNN and PNN gave convincing results of R Squared values on both the datasets i.e. QUES (GRNN R- squared value = 0.36, GRNN RMSE value = 41.52, PNN R- squared value = 0.4, PNN RMSE value = 39.07) and UIMS (GRNN R- squared value = 0.42, GRNN RMSE value = 34.17, PNN R- squared value = 0.45, PNN RMSE value = 31.5). Therefore, it was closed GRNN and PNN are without a doubt a helpful displaying procedure and it could be utilized as a sound option for the prediction of software maintainability.

1. INTRODUCTION

There are many definitions available in literature about ‘software maintainability’ and more or less all are similar. IEEE Standard Glossary of Software Engineering Terminology (IEEE 1990) defines ‘software maintainability’ [2, 3, 4] as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. Data gathered about software industry strongly indicates that almost 75 % of project cost is actually spent on maintenance. Maintenance phase starts once the software is delivered to the customer and during this phase changes made into software are inevitable. It not only affects developers in terms of cost and efforts associated but also determines how the customers perceived towards. Cost associated to maintain one line of source code is 10 times the cost of initial development of that line. Hence, software industry today strives hard to find ways and means to measure ‘‘Ease of maintainability’’ not only to reduce cost but also to ascertain whether maintenance of that product is worthwhile or not.

New programs nowadays are unpredictable and the size of programming has expanded significantly, in this manner making the product progressively hard to keep up. Programming viability is straightforwardly related with the monetary presentation and accomplishment of an item or an association. Utilizing viability [3, 4, 5], we can anticipate what changes that may happen in programming after it has been conveyed. Viability of the product is in this manner a quality property for programming that assists with deciding the exhibition of the product. Designers are helped to decide plausible measure of progress that may happen in programming modules during the upkeep stage. Attributable to the rightness of the practicality expectations, the structure of programming can be improved to decide changes that should be made for advancement of programming modules later on.

There have been many machine learning and artificial intelligence approaches that have been used till date. In majority of the approaches, only a small number of software has been used for testing. Indeed, datasets are not sufficient enough to give conclusive results which provide information regarding the input data of the system. The work regarding Software Maintainability Prediction [2, 3, 5, 9] is based on applying different machine learning algorithms on software models and providing a comparative study between the different software metrics and machine learning algorithms.

Machine Learning Techniques [17, 20] are one of the numerous procedures of AI that are accessible in writing. The contributions from the prior layers are utilized as the yield for every one of the progressive layers. The learning procedure can be completely regulated, semi-managed or solo, and this technique utilizes a chain of command idea where each layer learns the numerous degrees of portrayal that relate to various degrees of deliberation. The system learns and recollects the arrangement of information and in each layer, and in this manner figures out which information to recall and which information to overlook. Along these lines, toward the finish of the procedure, we can figure out which measurements will be added to software maintainability to make exact prediction.

Therefore in this project will focus on creating models using 9 machine learning algorithms to predict the software maintainability cost using the 2 widely used datasets – QUES and UIMS whose frameworks are written in classical ADA

2. PROBLEM STATEMENT

The goal of this project is to build machine learning models that can help us in predicting the software maintainability cost, based on a combination of features that describes the software so that the user saves the time and money in maintaining the software.

3. PROJECT BENEFITS

The software maintainability prediction model will help us in the following aspects:

(a). It helps in keeping future maintenance effort under control.

(b). It provides managers with information for more effectively planning the use of valuable resources.

(c). It helps managers in taking important decision regarding staff allocation.

(d). It guides about maintenance process efficiency.

(e). It enables the developers to identify the determinants of software quality so that they can improve design and coding.

(f). The threshold values of various metrics which drastically affect maintainability of software can be checked and kept under control so as to achieve least maintenance cost.

(g). It helps project managers in comparing the productivity and costs among different projects.

(h). It helps practitioners to improve the quality of software systems and thus optimize maintenance costs.

4. Literature Survey

Many surveys and examples have been presented in past to show that software design metrics are correlated with its maintainability.

N. Baskar and C. Chandrasekar [1] compares the three existing models namely GMDH, GRNN PNN with proposed Neuro-PSO (NPSO) on the basis of four measures: MRE, MMRE, Pred (0.25) and Pred (0.75) from which they concluded that out of the four models, Neuro-PSO gives the best results and the least value for MMRE and maximum values for Pred (0.25) and Pred (0.75)

Chug and Malhotra [3, 4] presented the prediction performance of the machine learning algorithms based models like GMDH, GA and PNN and assessed and compared with prevailing models in terms of MMRE, Pred(0.25) and Pred(0.30). They found that GMDH model outperformed the prevailing models as the least MMRE value is recorded. As far as Pred(0.25) and Pred(0.30) values. They even presented a study aimed at assessing the efficiency of different prediction models for prediction maintainability of web based systems using Object Oriented metrics. The results show that the GMDH is very helpful model in prediction of software maintainability. They made the aim of this study is not only to verify whether relationship between metrics and maintainability do exists in real life web based applications but also to verify supremacy of GMDH.

Fioravanti and Nesi [5] presented a metric analysis to identify which metrics would be better ranked for its impact on the prediction of adaptive maintenance for object-oriented systems. The model and metrics proposed have been validated against real data by using MLR (Multi linear Regression Analysis) Model. The validation has identified that the several metrics can be profitably employed for the prediction of software maintainability.

Dagpinar [12] also based their study on empirical data to establish the relationship between software metrics and its maintainability however instead of design level metrics of structure languages, the metrics were replaced by object oriented metrics. They recorded significant impact of two metrics i.e. direct coupling metric and size metric on software maintainability while other parameters like cohesion, inheritance and indirect coupling were not considered significant by them.

Zhou and Leung [22] have used multivariate adaptive regression splines (MARS) for predicting object-oriented software maintainability in 2007. They compared the prediction accuracy of proposed model with four other prevailing models: multivariate linear regression (MLR), support vector regression (SVR), artificial neural network (ANN), and regression tree (RT) and stated that MARS is best model to be used as far as maintainability of prediction is concerned

5. Research Methodology

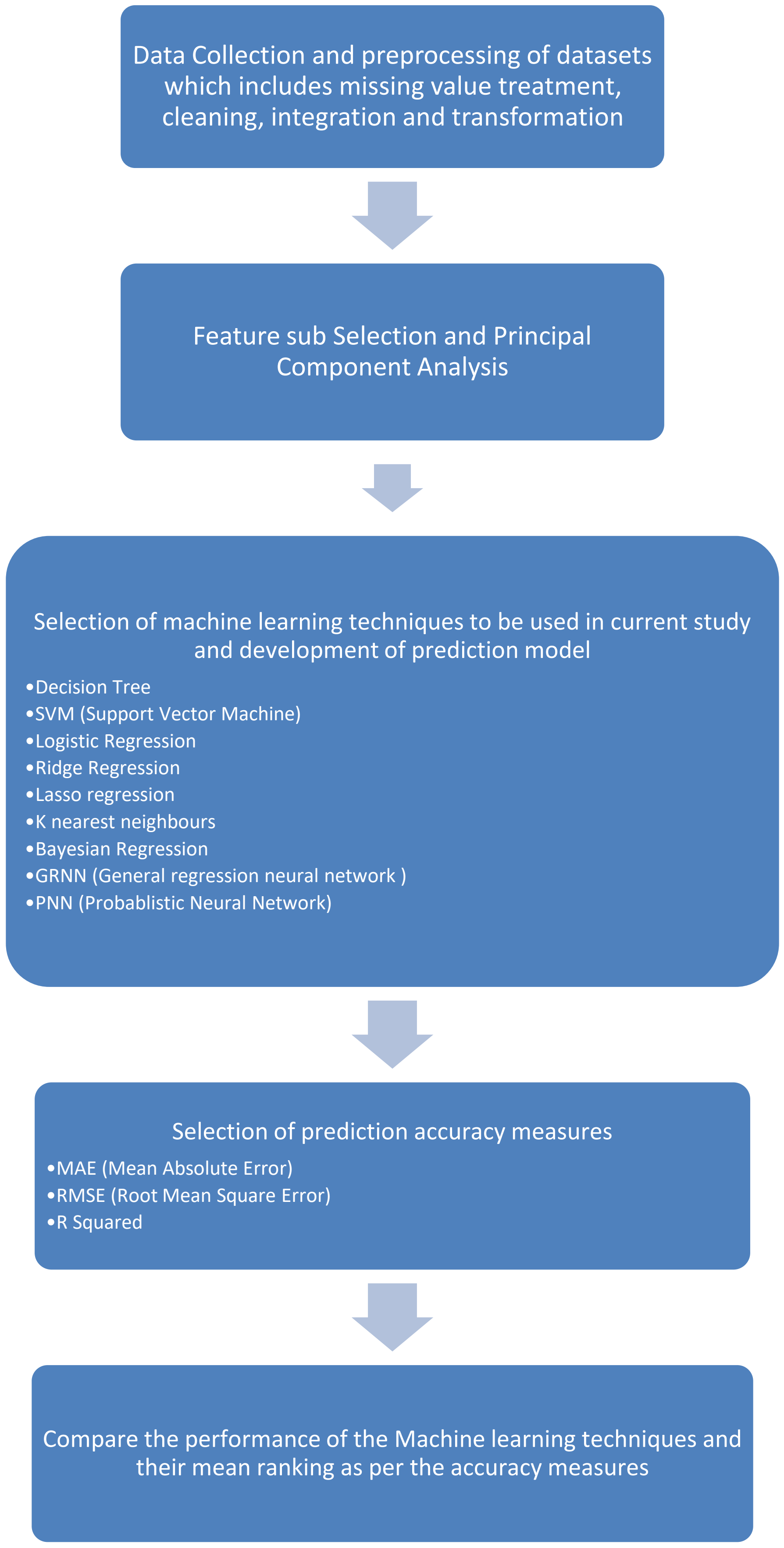
Data Collection and preprocessing of datasets which includes missing value treatment, cleaning, integration and transformation
Feature sub Selection and Principal Component Analysis
Selection of machine learning techniques to be used in current study and development of prediction model
•Decision Tree
•SVM (Support Vector Machine)
•Logistic Regression
•Ridge Regression
•Lasso regression
•K nearest neighbours
•Bayesian Regression
•GRNN (General regression neural network)
•PNN (Probablistic Neural Network)
Selection of prediction accuracy measures
•MAE (Mean Absolute Error)
•RMSE (Root Mean Square Error)
•R Squared
Compare the performance of the Machine learning techniques and their mean ranking as per the accuracy measures

Figure 1- Methodology flowchart

The following is the description of the research methodology we have used in Fig:1

5.1 Approaches

In this section we have given brief introduction of all machine learning models which are used in current study.

5.1.1. Programming Language

5.1.1.1. Python

This project will be made of Python [20] which is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

5.1.2. Feature Selection

5.1.2.1. Cross validation

Cross-validation [21] is a technique in which the model is trained using the subset of the data-set and then evaluates using the complementary subset of the data-set.

The three steps involved in cross-validation are as follows:

- Reserve some portion of sample data-set.
- Using the rest data-set train the model.
- Test the model using the reserve portion of the data-set

5.1.2.2. Principal Component Analysis

Principal Component Analysis [21] is basically a statistical procedure to convert a set of observation of possibly correlated variables into a set of values of linearly uncorrelated variables.
Each of the principal components is chosen in such a way so that it would describe most of the still available variance and all these principal components are orthogonal to each other. In all principal components first principal component has maximum variance.

5.1.3. Machine Learning Algorithms

5.1.3.1. Decision Tree

Due to the working for both categorical and continuous input and output variables this project will be using decision tree algorithm in our project. In this technique, a split of population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables. Decision tree [21] is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems.

5.1.3.2. Support Vector Machine
“Support Vector Machine” (SVM) [21] is a supervised machine learning algorithm which will be used in our project as it can be used for either classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, perform classification by finding the hyper-plane that differentiate the two classes very well.

5.1.3.3. Logistic Regression
Logistic regression [20] is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

5.1.3.4. K nearest Neighbors
K nearest neighbors [18] is a simple algorithm that stores all available cases and predicts the numerical target based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970’s as a non-parametric technique. A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. Another approach uses an inverse distance weighted average of the K nearest neighbors. KNN regression uses the same distance functions as KNN classification.

5.1.3.5. Ridge regression
Ridge regression [21] addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

$$\text{minw } ||Xw-y||22+\alpha||w||22 \qquad (1)$$

The complexity parameter $\alpha \geq 0$ controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to co linearity.

5.1.3.6. Lasso Regression

The Lasso [21] is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent. For this reason Lasso and its variants are fundamental to the field of compressed sensing. Under certain conditions, it can recover the exact set of non-zero coefficients. Mathematically, it consists of a linear model with an added regularization term. The objective function to minimize is:

$$\text{minw12nsamples}||Xw-y||22+\alpha||w||1 \qquad (2)$$

The lasso estimate thus solves the minimization of the least-squares penalty with $\alpha||w||1$ added, where α is a constant and $||w||1$ is the $\ell1$-norm of the coefficient vector.

5.1.3.7. Bayesian Regression

Bayesian regression [21] techniques can be used to include regularization parameters in the estimation procedure: the regularization parameter is not set in a hard sense but tuned to the data at hand. This can be done by introducing uninformative priors over the hyper parameters of the model. The $\ell 2$ regularization used in Ridge regression and classification is equivalent to finding a maximum a posteriori estimation under a Gaussian prior over the coefficients w with precision $\lambda-1$. Instead of setting lambda manually, it is possible to treat it as a random variable to be estimated from the data. To obtain a fully probabilistic model, the output y is assumed to be Gaussian distributed around Xw:

$$p(y|X,w,\alpha)=N(y|Xw,\alpha) \tag{3}$$

5.1.3.8. General Regression Neural Network

The general regression neural network (GRNN) [19] is a single-pass neural network which uses a Gaussian activation function in the hidden layer. GRNN consists of input, hidden, summation, and division layers. The regression of the random variable y on the observed values XX of random variable xx can be found using

$$E[y|X]=\int\infty-\infty\ yf(X,y)dy\ /\ \int\infty-\infty\ f(X,y)dy \tag{4}$$

where f(X,y) is a known joint continuous probability density function. When f(X,y) is unknown, it should be estimated from a set of observations of x and y. f(X,y) can be estimated using the nonparametric consistent estimator suggested by Parzen

5.1.3.9. Probabilistic Neural Network (PNN)

A probabilistic neural network (PNN) [20] is a feed forward neural network, which is widely used in classification and pattern recognition problems. In the PNN algorithm, the parent probability distribution function (PDF) of each class is approximated by a Parzen window and a non-parametric function. Then, using PDF of each class, the class probability of a new input data is estimated and Bayes' rule is then employed to allocate the class with highest posterior probability to new input data. By this method, the probability of mis-classification is minimized. This type of ANN was derived from the Bayesian network and a statistical algorithm called Kernel Fisher discriminant analysis. It was introduced by D.F. Specht in 1966. In a PNN, the operations are organized into a multi-layered feedforward network with four layers:

- Input layer
- Pattern layer
- Summation layer
- Output layer

5.2 Dataset

First the metrics of the dataset is described, which are as follow:

Table 1: Metrics Description

WMC (Weighted Methods Per Class)	The sum of McCabe's cyclomatic complexities of all local methods in a class. Let a class K1 with method M1…… Mn that are defined in the class. Let C1…….Cn be the complexity of the methods. Which can be written as: $$WMC = \sum_{i=1}^{n} Ci$$
DIT (Depth of Inheritance Tree)	The depth of a class in the inheritance tree where the root class is zero.
NOC (Number of Children)	The number of child classes for a class. It counts number of immediate sub classes of a class in a hierarchy.
RFC (Response for a Class)	The number of local methods plus the number of non local methods called by local methods.
LCOM (Lack of Cohesion of Methods)	The number of disjoint sets of local methods. Each method in a disjoint set shares at least one instance variable with at least one member of the same set.
MPC (Message Passing Coupling)	The number of messages sent out from a class.
DAC (Data Abstraction Coupling)	The number of instances of another class declared within a class.
NOM (Number of Methods)	The number of methods in a class.
SIZE1 (Lines of code)	The number of lines of code excluding comments.
SIZE2 (Number of properties)	The total count of the number of data attributes and the number of local methods in a class.
CHANGE (Number of lines changed)	The number of lines added and deleted in a class change of the content is counted as two.

In our project the use two most popular object oriented maintainability datasets which are also published by Li and Henry [2] : UIMS and QUES datasets. Their descriptive statistics is given in table followed by the interpretation. These datasets were chosen mainly because they have been recently used by many researchers to evaluate the performance of their proposed model in predicting object-oriented software maintainability [6, 8, 9, 11] and hence are wanted to be able to compare our results against this published work. 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. Both systems were implemented in Ada. Both datasets consist of eleven class-level metrics: ten independent variables and one dependent variable. The independent variables are taken as follows:

(i) Five variables are taken from Chidambar et al. [14] : WMC, DIT, NOC, RFC, and LCOM;
(ii) Four variables are taken from Li and Henry [2, 13]: MPC, DAC, NOM, and SIZE2;
(iii) One variable is taken from traditional lines of code metric (SIZE1).
(iv) The dependent variable is a maintenance effort surrogate measure (CHANGE), which is the number of lines in the code that were changed per class during a 3-year maintenance period. 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.

Table 2: Descriptive Statistics of UIMS dataset

Metric	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
WMC	0	69	11.38	15.90
DIT	0	4	2.15	0.90
NOC	0	8	0.95	2.01
RFC	2	101	23.21	20.19
LCOM	1	31	7.49	6.11
MPC	1	12	4.33	3.41
DAC	0	21	2.41	4.00
NOM	1	40	11.38	10.21
SIZE1	4	439	106.44	114.65
SIZE2	1	61	13.47	13.47
CHANGE	2	253	42.46	61.18

Table 3: Descriptive Statistics of QUES dataset

Metric	Minimum	Maximum	Mean	Standard Deviation
WMC	1	83	14.96	17.06
DIT	0	4	1.92	0.53
NOC	0	0	0.00	0.00
RFC	17	156	54.44	32.62
LCOM	3	33	9.18	7.31
MPC	2	42	17.75	8.33
DAC	0	25	3.44	3.91
NOM	4	57	13.41	12.00
SIZE1	115	1009	275.58	171.60
SIZE2	4	82	18.03	15.21
CHANGE	6	217	64.23	43.13

From the descriptive statistics [3] we noticed some observations and accordingly actions were taken. Some of them are mentioned as follows:

(i) For DIT Median and Mean value are minimum in both the system, so a conclusion is drawn that the use of inheritance in both systems is limited.

(ii) The values for median and mean for CHANGE (dependent variable) in the UIMS dataset is lesser than those in the QUES, which means UIMS seems to be more maintainable.

(iii) We had removed NOC from the QUES dataset because it was observed that all data points for NOC are zeros in the QUES dataset.

(iv) We had observed that the coupling between classes in QUES was higher than those in the UIMS because the medians and means values for RFC and MPC in the QUES dataset were larger than UIMS dataset.

(v) Values of Mean and Median of LCOM were almost same in both systems that mean both have almost similar cohesion.

(vi) The similar medians and means for NOM and SIZE2 in both datasets suggest that both systems had similar class sizes at the design level. However, there was a significant difference in SIZE1.

5.3 Prediction Accuracy Measures

5.3.1 Mean Squared Error

The measure of mean squared error [20] needs a target of prediction or estimation along with a predictor or estimator, which is said to be the function of the given data. MSE is the average of squares of the “errors”. Here, the error is the difference between the attribute which is to be estimated and the estimator. Let us suppose that X_i is the vector denoting values of n number of predictions. Also, X_i is a vector representing n number of true values. Then, the formula for mean squared error is given below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2 \quad (5)$$

5.3.2 Root Mean Square Error Formula

The root mean square error (RMSE) [20] is a very frequently used measure of the differences between value predicted value by an estimator or a model and the actual observed values. RMSE is defined as the square root of differences between predicted values and observed values. The individual differences in this calculation are known as “residuals”. The RMSE estimates the magnitude of the errors. It is a measure of accuracy which is used to perform comparison forecasting errors from different estimators for a specific variable, but not among the variables, since this measure is scale-dependent.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2} \quad (6)$$

5.3.3 Mean Absolute Error

In statistics, mean absolute error (MAE) [20] is a measure of errors between paired observations expressing the same phenomenon. Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as:

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n} | \boldsymbol{x}_i - \boldsymbol{x} | \quad (7)$$

It is thus an arithmetic average of the absolute errors , where is the prediction and the true value. Note that alternative formulations may include relative frequencies as weight factors. The mean absolute error uses the same scale as the data being measured

5.3.4 R squared error

In statistics, the coefficient of determination, denoted R^2 or r^2 and pronounced "R squared" [20], is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2} \quad (8)$$

Sum of squared errors of our regression model (SSres) Sum of squared errors (SStot)

6. Results and Discussion

6.1 Results

Comparison of various models with reference to their predictive performance for QUES and UIMS system are provided in table 4.

Table 4- Comparison of various models with reference to their predictive performance for QUES and UIMS system

Model	UIMS - R squared	QUES - R squared	UIMS - RMSE	QUES - RMSE	UIMS - MAE	QUES - MAE
GRNN	0.42	0.36	34.17	41.52	20.75	28.06
PNN	0.45	0.40	31.50	39.07	20.50	27.47
Decision Tree	0.32	0.31	23.63	38.94	13.38	25.13
Decision Tree with PCA	0.19	0.21	25.77	41.52	15.38	28.06
SVM	0.32	0.19	23.63	42.40	13.38	27.47
SVM with PCA	0.14	0.09	26.46	49.18	18.80	37.06
Logistic Regression	0.15	0.48	58.34	33.84	31.25	22.60
Logistic Regression with PCA	0.29	0.20	24.11	46.27	11.13	32.60
KNN	0.09	0.64	27.29	28.09	24.12	16.77
KNN with PCA	0.47	0.31	20.78	39.07	16.01	28.21
Ridge	0.10	0.64	28.49	27.88	19.14	20.71
Ridge with PCA	0.35	0.46	25.63	34.43	14.38	25.24
Lasso	0.17	0.59	26.16	29.93	17.04	21.69
Lasso with PCA	0.26	0.46	46.91	34.48	33.62	25.30
Bayesian	0.09	0.55	56.96	31.19	39.51	22.51
Bayesian with PCA	0.27	0.43	47.31	35.18	34.14	26.59

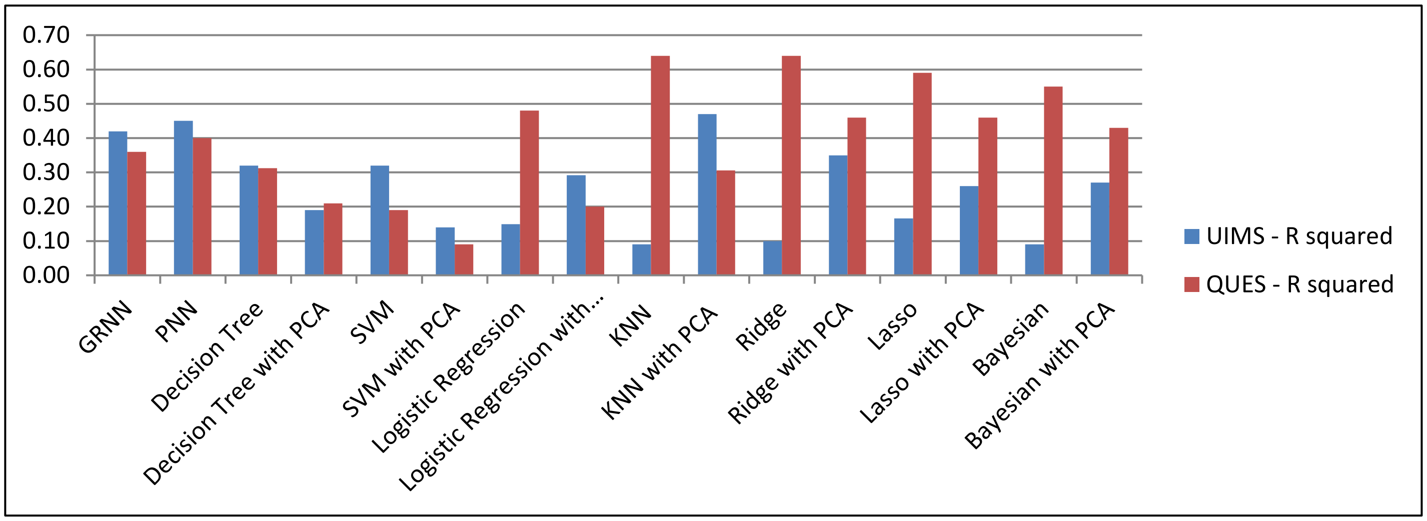

Figure 2- R Squared values of Proposed Models

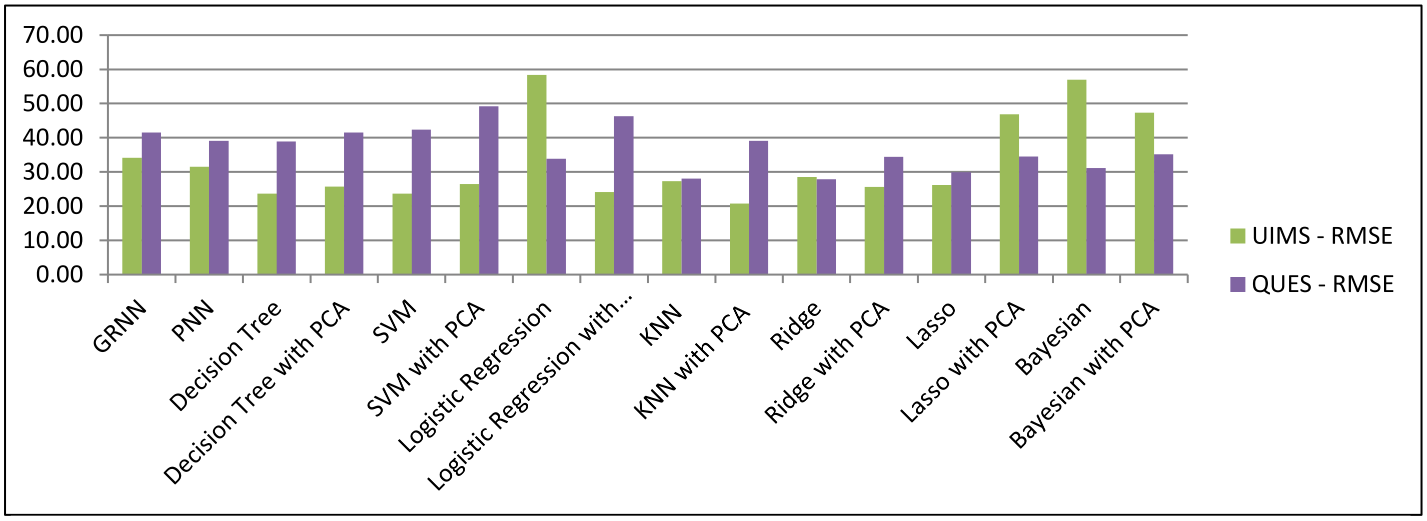

Figure 3- RMSE values of Proposed Models

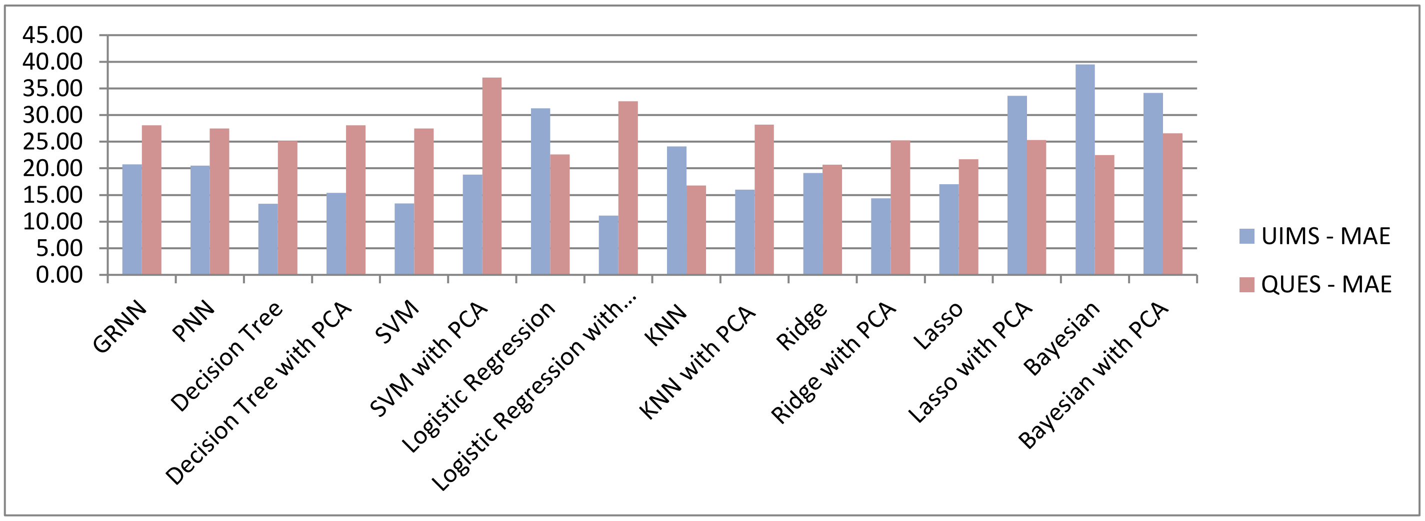

Figure 4- MAE values of Proposed Models

7. Results and Analysis

This paper has presented an empirical study that sought to build object-oriented software maintainability prediction model using following nine machine learning algorithms: Decision Tree, Support Vector Machine, K-nearest Neighbors, Logistic Regression, Ridge Regression, Lasso Regression, Bayesian Regression, General Regression Neural Network and Probabilistic Neural Network.

It can be seen in Figure 2, Figure 3, and Figure 4 that after applying Principal Component Analysis with the model the results were not getting any better when compared to the results of the models without the principal component analysis.

The models worked better and gave better results of R Squared values with the QUES dataset whereas the other prediction accuracy measures gave better results of the models with the UIMS dataset as seen in Figure 2, Figure 3, and Figure 4.

To analyze the prediction accuracy measure we have taken R squared values and RMSE values of all models and presented them.

It can be observed that out of the nine algorithms selected and evaluated, the KNN (R- squared value = 0.64, RMSE value = 28.09) and Ridge regression (R- squared value = 0.64, RMSE value = 27.88) model gives very competitive results on the QUES dataset.

But the GRNN and PNN models has given moderate results on both the datasets i.e. QUES (GRNN R-squared value = 0.36, GRNN RMSE value = 41.52, PNN R- squared value = 0.4, PNN RMSE value = 39.07) and UIMS (GRNN R- squared value = 0.42, GRNN RMSE value = 34.17, PNN R- squared value = 0.45, PNN RMSE value = 31.5) and hence show their worth that they can be used in the process of software maintainability prediction.

8. Conclusion and Future Direction

Nine distinctive machine learning models are utilized with the end goal of expectation of programming practicality. Despite the fact that numerous examinations detailed wide use of GMDH model and GA model in various fields with the end goal of forecast of high request input yield relationship which is intricate, non direct and unstructured yet just because they are utilized for expectation of programming practicality.

The objective of our investigation is to build appropriate model utilizing machine learning algorithms for the forecast of item arranged programming practicality which are anything but difficult to apply as well as could lessen the expectation mistakes to least.

The prediction performance of the machine learning algorithms based models were assessed and compared with prevailing models in terms of R Squared values, MAE values and RMSE values. It was found that KNN (R- squared value = 0.64, RMSE value = 28.09) and Ridge regression (R- squared value = 0.64, RMSE value = 27.88) model outperformed the prevailing models as the least RMSE value is recorded. As far as R Squared values are concerned, KNN and Ridge Regression models are significantly better over others on the QUES dataset whereas GRNN and PNN gave compelling results of R Squared values on both the datasets i.e. QUES (GRNN R- squared value = 0.36, GRNN RMSE value = 41.52, PNN R- squared value = 0.4, PNN RMSE value = 39.07) and UIMS (GRNN R- squared value = 0.42, GRNN RMSE value = 34.17, PNN R- squared value = 0.45, PNN RMSE value = 31.5). Thus, it is concluded GRNN and PNN are indeed a very useful modeling technique and it could be used as a sound alternative for the prediction of software maintainability.

As the proposed model was discovered appropriate for assessing the product dependability in a others work, accordingly with current discoveries it very well may be securely assumed that both the unwavering quality and viability, which stay irreplaceable parts of programming quality, can be anticipated with use of same model for example GRNN. This will unquestionably decrease the difficulties engaged with forecast of viability and helps programming designers to deliberately use their assets, improve process proficiency and enhance the related upkeep costs.

The present investigation depended on two business programming datasets UIMS and QUES created in ADA, in this manner we can effectively engage in doing additionally work that applies the proposed GRNN and PNN arrange model to progressively objective datasets to learn its credibility for more extensive programming ideal models. Such examinations would permit us to research the capacity of models lastly setting up a summed up model in the field of Software Quality. Further research can be arranged trying to join these models with other information mining procedures to create expectation models which can gauge the viability of programming all the more precisely with least accuracy mistakes on other programming dialects.

9. References

[1]. N. Baskar and C. Chandrasekar “An Evolving Neuro-PSO-based Software Maintainability Prediction”

[2]. W. Li and S. Henry “Object-Oriented Metrics that Predict Maintainability”

[3]. Ruchika Malhotra and Anuradha Chug “Software Maintainability Prediction using Machine Learning Algorithms”

[4]. Ruchika Malhotra and Anuradha Chug “Application of Evolutionary Algorithms for Software Maintainability Prediction using Object-Oriented Metrics”

[5]. F. Fioravanti and P. Nesi "Estimation and prediction metrics for adaptive maintenance effort of object oriented systems”

[6]. KK Aggarwal, Y Singh, JK Chhabra “An Integrated Measure of Software Maintainability”

[7]. Ruchika Malhotra and Anuradha Chug “Application of Group Method of Data Handling model for software maintainability prediction using object oriented systems” Ruchika Malhotra and Anuradha Chug

[8]. M. Thwin and T. Qua "Application of neural networks for software quality prediction using object oriented metrics”

[9]. K.K. Aggarwal, Y. Singh, P. Chandra and M. Puri “Measurement of Software Maintainability Using a Fuzzy Model”

[10]. Anuradha Chug and Ruchika Malhotra “Benchmarking framework for maintainability prediction of open source software using object oriented metrics”

[11]. “An application of Bayesian network for predicting object-oriented software maintainability” C.V. Koten, A.R. Gray

[12]. M. Dagpinar and J.H. Jahnke “Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison”

[13]. W. Li “Another Metric Suite for Object-oriented Programming”

[14]. S. Chidamber and C. Kemerer "A Metrics Suite for Object Oriented Design,"

[15]. Ruchika Malhotra and Anuradha Chug ”Application of Group Method of Data Handling model for software maintainability prediction using object oriented systems”

[16]. Ruchika Malhotra and Anuradha Chug “A metric suite for predicting software maintainability in data intensive application”

[17]. Deeksha Datyal and Aman Kaushik "A Systematic Review of Software Maintainability Prediction"

[18]. https://www.saedsayad.com/k_nearest_neighbors_reg.html

[19].https://www.intechopen.com/books/digital-systems/applications-of-general-regression-neural-networks-in-dynamic-systems

[20]. https://en.wikipedia.org

[21]. https://scikit-learn.org/stable/modules

[22]. Zhou Y and Leung H "Predicting object-oriented software maintainability using multivariate adaptive regression splines."