Vol. XVI & Issue No. 11 November - 2023
INDUSTRIAL ENGINEERING JOURNAL

# SOFTWARE MAINTAINABILITY PREDICTION FOR OBJECT - ORIENTED SYSTEMS USING DEEP LEARNING

**Anita Devi**
**Saurabh Charaya**
**Mukesh Maan**

**Abstract**

*Lots of effort has been used to minimize the software maintenance level of any software specially developed with object-oriented methodology. In this paper, we have proposed a software maintainability prediction model for object-oriented systems using Deep learning. Google Collaboratory has been used with UQES and UIMS datasets for experimental purposes. MAE, MSE, and Varscore have been used as performance metrics. From the results, it was found that varscore for both datasets increases with an increase in the number of hidden layers. However, For the UIMS dataset, the accuracy of the model is 93.7% whereas, for the QUES dataset, the accuracy of the model is 73.12%.*

***Keywords: Software Maintainability, Object-Oriented Systems, Neural Network, Deep Neural Network, MAE and MSE.***

## 1. INTRODUCTION

Software Maintenance is one of the utmost parts of the Software Development Life Cycle and is considered to be sometimes more expensive than even software development. For any software product, Maintenance behavior is very difficult to track. Software Maintenance requires a huge amount of time and training resources. Accurate Maintenance prediction helps managers to take essential decisions and improve the quality of software. Numerous attributes such as coupling, cohesion, change, maintainability index etc. have been used in the past to forecast the maintenance behavior of Object-Oriented systems.

Li W and Henry S (1993) investigated the relationship between metrics and software maintainability prediction for object-oriented systems and it was concluded that there is a strong relationship between metrics and maintenance effort. Further, maintenance effort can be effectively calculated from the combination of metrics. QUES, UIMS, NASA, UML Class Diagram, Apache Lucene etc. are some of the prevalent datasets used for software Maintainability Prediction. The use of Soft Computing techniques for predicting the maintainability of software is quite limited due to the high computational cost, convergence and privacy issues (Yenduri G and Gadekallu T R (2022)). Machine learning techniques are also being very widely used these days for predicting the maintenance of software.

In this research paper, deep learning a subset of machine learning techniques has been used for predicting the software's maintainability. Deep Learning comprises neural networks with three or more layers and is gaining immense popularity these days because of its ability to learn better relationships. MAE, MSE, and Varscore have been used to calculate the effectiveness of the proposed model. Section 2 describes the literature review. Experimental setup and implementation of the proposed model have been described in section 3 and section 4 respectively.

## 2. LITERATURE REVIEW

A systematic review has been performed by Malhotra R and Chug A (2016) for the current trends in software maintainability and it was found that Machine Learning and Evolutionary algorithms have been used widely in this field and design metrics are most popular for capturing the dependencies of the software. Jha S et al. (2019) implemented five Machine Learning algorithms i.e. Ridge Regression, Decision Trees, Quantile Regression Forest, Support Vector Machines, and Principal Component Analysis for predicting the software's maintainability and it was found that the deep learning technique performed better. Alsolai S and Roper R (2020) performed a review of Machine Learning techniques for predicting the software's maintainability and it was found that the ensemble models provided higher prediction accuracy than individual models.

Kumar V et al. (2014) built an ANN model using input factors such as the count of multiple conditions, count of nodes, percentage comments, and total lines of code has been used for maintainability prediction. The model is evaluated on the historical data in terms of RMSE and the results proved the efficiency of the proposed model. Kumar L and Rath S K (2017) implemented the Neuro Fuzzy model for building a maintainability model using UIMS and QUES datasets. Rough Set Analysis (RSA) and Principal Component Analysis (PCA) have been implemented for selecting the subset of metrics from 10 available inputs and it was interpreted that the selected metrics have improved accuracy for forecasting the maintainability.

Malhotra R and Chug A (2012) implemented the Group Method of Data Handling (GMDH), Genetic Algorithms (GA), and Probabilistic Neural Network (PNN) for maintainability prediction using UIMS and QUES datasets. The performance of the proposed models has been compared with the existing techniques and the results concluded that GMDH performed better. Alsolai H et al (2018) compared the performance of

individual models and bagging ensemble models for forecasting the maintainability of software using the QUES dataset and it was found that bagging ensemble models with k-nearest neighbors achieved superior performance.

Henry S et al. (1990) determined the system's maintainability for Object Oriented and Procedural languages and it was interpreted that systems developed using Object Oriented languages are more maintainable. Dhaka V P and Dhaka S (2013) predicted the software maintainability using the Machine learning approach on the QUES dataset and it was found that Gaussian process regression networks (GPRN) performed better.

Gupta S and Chug A (2021) performed a study of Machine Learning based Boosting Algorithms (BAs) for predicting the software's maintainability and it was found that BA algorithms performed superior. Gopal M K and Amirthavalli M (2019) implemented ML algorithms to identify metrics for software applications using Object Oriented methodology and it was found that coupling is the most important contributing factor. Gupta S and Chug A (2020) implemented the Least Squares Support Vector Machines (LS-SVM) algorithm for SM Prediction (SMP) on six datasets and the results indicated that the LS-SVM performed better.

Elmidaoui S et al. (2020) conducted an experiment on the accuracy analysis of Machine Learning Techniques and it found that Support Vector Machines, Decision Trees and Neuro-Fuzzy Techniques provide more accurate results in terms of prediction and Mean Magnitude of Relative Error (MMRE). Dagpinar M and Weber J (2003) analyzed numerous metrics to determine the noteworthy metrics for predicting the maintainability of software and it was concluded that size and import direct coupling are the substantial metrics. Şahin C B (2021) developed a deep learning-based model for predicting software maintainability using vulnerable software metrics and the proposed model was found to be accurate and effective.

## 3. EXPERIMENTAL SETUP

In this research work, UIMS and QUES datasets have been used for experimental purposes. UIMS and QUES dataset comprises 10 input attributes and 71 rows whereas the QUES dataset comprises 10 attributes and 39 rows. Deep Neural Networks have been implemented using Google Colaboratory.

## 4. IMPLEMENTATION

For implementing Deep Neural Networks, the first step is to install Java and Keras into Google Colaboratory. The code for the same has been shown below in Figure 1. Input variables for the neural network have been specified in Table 1 and the output is to predict the change i.e. maintenance effort. Change is determined by the number of lines changed per class in its maintenance history. The architecture for the Deep Neural Network has been shown in Figure 2.

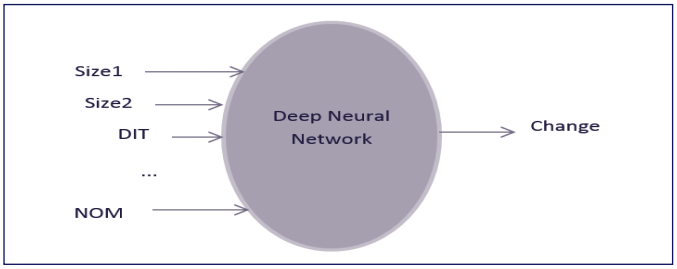
**Table 1: Input attributes for Deep Neural Network**

| Input Variables | Significance |
|---|---|
| Size 1 | Number of semi-colons per class |
| Size2 | Sum of number of methods and attributes |
| DIT | Depth in the inheritance tree |
| NOC | Number of children |
| MPC | Message-passing coupling |
| RFC | Response for class |
| LCOM | Lack of cohesion of methods |
| DAC | Data abstraction coupling |
| WMC | Weighted method complexity |
| NOM | Number of methods |

**Figure 1: Code for installation of Java and Keras in Google Collaboratory**

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!pip install -q keras
```

**Figure 2: Architecture of the Deep Neural Network for Software Maintenance Prediction**




As all the input attributes have the varying range, thus next step is to scale all the input attributes in the range of 0 and 1 which is achieved by using MinMaxScalar and the code has been shown in Figure 3. After data pre-processing, dataset is split into training and test set in the ratio 80:20.

**Figure 3: Python Code for scaling the input attributes**

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
print(x_scaled)
```

Keras Sequential model is then used to build the Deep Neural Networks. The number of layers in the network has been increased from 1 to 5 and its impact on the Mean Square Error (MSE) of the network has been observed. The code for the network has been described in Figure 4. Input Layer of the network has 10 neurons as there are 10 input attributes while the output layer has 1 neuron for predicting the maintenance. ReLU activation function has been used in hidden layers of the network. MSE loss function and Adam optimizer has been used for training of network. Figure 5 describes the summary of the Deep Neural Net having 5 hidden layers.

**Figure 4: Python Code for Building the Neural Network**

```
model = Sequential()
model.add(Dense(10, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(1, name="predictions"))
```

**Figure 5: Summary of the Deep Neural Network comprising 5 hidden layers**

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_17 (Dense)            (None, 10)                110

 dense_18 (Dense)            (None, 100)               1100

 dense_19 (Dense)            (None, 100)               10100

 dense_20 (Dense)            (None, 100)               10100

 dense_21 (Dense)            (None, 100)               10100

 dense_22 (Dense)            (None, 100)               10100

 predictions (Dense)         (None, 1)                 101

=================================================================
Total params: 41,711
Trainable params: 41,711
Non-trainable params: 0
_________________________________________________________________
```

## 5. PERFORMANCE METRICS

To evaluate the performance of the model, follows metrics have been used:

a) Mean Absolute Error (MAE): MAE evaluates the average of the difference between the real and predicted values of all the observations

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i^{real} - y_i^{pred}|$$

b) Mean Squared Error (MSE): It evaluates the average of the squares of the errors.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i^{real} - y_i^{pred})^2$$

c) Varscore: It is similar to $R^2$ but it does not account for systematic offsets in the prediction.

## 6. EXPERIMENTAL RESULTS

Table 2 and Table 3 describe the performance of a Deep Neural Network with up to 5 hidden layers for QUES and UIMS datasets respectively. Figure 6 and Figure 7 describes MAE and MSE values for both UIMS and QUES dataset with an increase in the number of hidden layers. From the results, it can be concluded that with an increase in the number of hidden layers in the network, MSE and MAE value decreases. Figure 8 describes the varscore of UIMS and QUES datasets. From the results, it can be interpreted that varscore for both datasets increases with an increase in the number of hidden layers. For the UIMS dataset, the accuracy of the model is 93.7% (with 5 hidden layers) whereas for the QUES dataset, the accuracy of the model is 73.12% (with 5 hidden layers).

**Table 2: Performance of Deep Neural Network for QUES dataset**

| No. of hidden layers | MAE | MSE | Varscore |
|---|---|---|---|
| 1 hidden layer | 26.34 | 822.9 | 26.46 |
| 2 hidden layer | 18.98 | 480.01 | 48.38 |
| 3 hidden layer | 14.85 | 364.49 | 63.34 |
| 4 hidden layer | 13.24 | 396.96 | 55.41 |
| 5 hidden layer | 10.5 | 239.41 | 73.12 |

**Table 3: Performance of Deep Neural Network for UIMS dataset**

| No. of hidden layers | MAE | MSE | Varscore |
|---|---|---|---|
| 1 hidden layer | 42.74 | 7648.5 | 6.03 |
| 2 hidden layer | 33.87 | 1833.42 | 73.34 |
| 3 hidden layer | 21.43 | 1043.47 | 84.42 |
| 4 hidden layer | 15.62 | 537.31 | 91.64 |
| 5 hidden layer | 13.47 | 403.29 | 93.7 |

**Figure 6: MAE value for UIMS and QUES dataset**

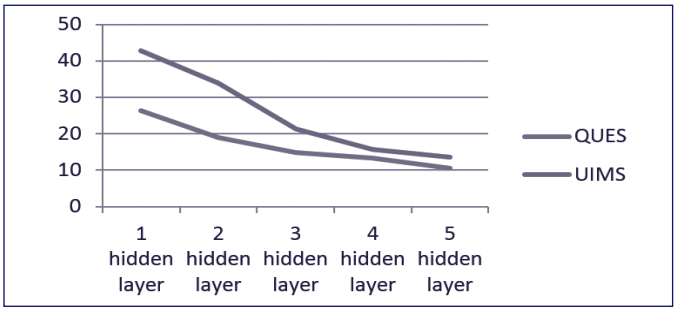



**Figure 7: RMSE value for UIMS and QUES dataset**

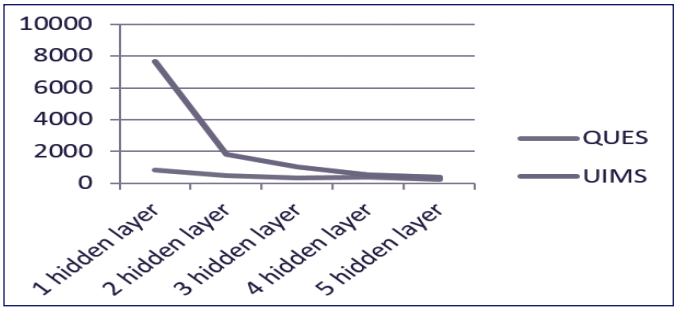
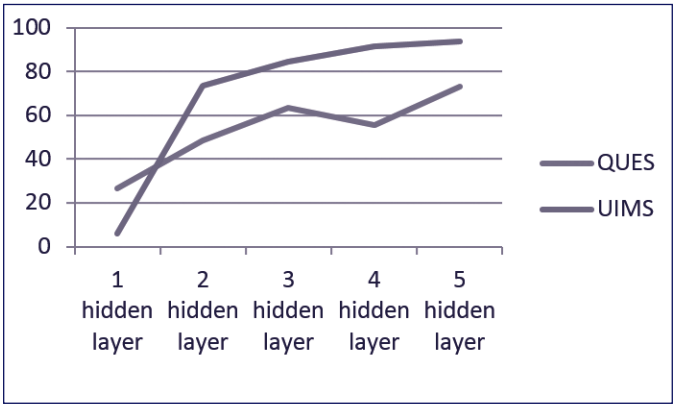



**Figure 8: Varscore for UIMS and QUES dataset**



## 7. CONCLUSION

Software Maintenance is one of the utmost parts of the Software Development Life Cycle. In this paper, Deep Neural Networks have been explored to predict the maintenance behavior of the software. Neural Networks with up to 5 hidden layers have been implemented using Google Colaboratory and its performance has been observed in terms of MSE, MAE, and varscore. From the results, it can be concluded that Deep Neural Network with 5 hidden layers gave the least MAE and MSE for both UIMS and QUES datasets.

## REFERENCES

1. *Elmidaoui S, Cheikhi L, Idri A and Abran A (2020), "Machine Learning Techniques for Software Maintainability Prediction: Accuracy Analysis", Journal of Computer Science and Technology, Vol. 35, pp1147–1174. https://doi.org/10.1007/s11390-020-9668-1*
2. *Jha S, Kumar R, Son L H, Abdel-basset M, Priyadarshini I, Sharma R and Long H V (2019), "Deep Learning Approach for Software Maintainability Metrics Prediction", IEEE Access, Vol. 7, pp61840-61855, 2019. https://doi.org/10.1109/ACCESS.2019.2913349*
3. *Alsolai H and Roper M (2020), "A Systematic Literature Review of Machine Learning Techniques for Software Maintainability Prediction", Information and Software Technology, Vol. 119. https://doi.org/10.1016/j.infsof.2019.106214*
4. *Kumar V, Kumar R and Sharma A (2014), "Maintainability Prediction from Project Metrics Data Analysis Using Artificial Neural Network: An Interdisciplinary Study", Middle-East Journal of Scientific Research, Vol. 19, No. 10, pp1412-1420.*
5. *Kumar L and Rath SK (2017), "Software Maintainability Prediction using Hybrid Neural Network and Fuzzy Logic Approach with Parallel Computing Concept", International Journal of System Assurance Engineering and Management. https://doi.org/10.1007/s13198-017-0618-4*
6. *Malhotra R and Chug A (2012), "Software Maintainability Prediction using Machine Learning Algorithms", Software Engineering: an International Journal (SeiJ), Vol. 2, No. 2.*
7. *Alsolai H, Roper M and Nassar D (2018), "Predicting Software Maintainability in Object-Oriented Systems Using Ensemble Techniques", International Conference on Software Maintenance and Evolution (ICSME), IEEE. https://doi.org/10.1109/ICSME.2018.00088*
8. *Henry S, Humphrey M and Lewis J (1990), "Evaluation of the Maintainability of Object-Oriented Software", IEEE Region 10 Conference on Computer and Communication Systems. Conference Proceedings, IEEE. https://doi.org/10.1109/TENCON.1990.152642*
9. *Dhaka V P and Dhaka S (2013), "Machine Learning Algorithm Evaluate the Maintainability", International Journal Of Computers & Technology, Vol. 10, No. 2, pp1376-1383. https://doi.org/10.24297/ijct.v10i2.7008*
10. *Gupta S and Chug A (2020), "An Extensive Analysis of Machine Learning Based Boosting Algorithms for Software Maintainability Prediction", International Journal of Interactive Multimedia and Artificial Intelligence, Vol. 7, No. 2.*
11. *Gopal M K and Amirthavalli M (2019), "Applying Machine Learning Techniques to Predict the Maintainability of Open Source Software", International Journal of Engineering and Advanced Technology (IJEAT), Vol. 8, No. 5S3. https://doi.org/10.35940/ijeat.e1045.0785s319*
12. *Gupta S and Chug A (2020), "Software Maintainability Prediction of Open Source Datasets using Least Squares Support Vector Machines," Journal of Statistics and Management Systems, Vol. 23, No. 6, pp1011-1021. https://doi.org/10.1080/09720510.2020.1799501*
13. *Dagpinar M and Weber J (2003), "Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison", Reverse Engineering - Working Conference Proceedings. pp 155- 164. https://doi.org/10.1109/WCRE.2003.1287246.*
14. *Li W and Henry S (1993), "Object-oriented Metrics that Predict Maintainability", Journal of Systems and Software, Vol. 23, No. 2, pp111-122. https://doi.org/10.1016/0164-1212(93)90077-B*
15. *Yenduria G and Gadekallu T R (2022), "A Systematic Literature Review of Soft Computing Techniques for Software Maintainability Prediction: State-of-the-Art, Challenges and Future Directions", Soft Computing Techniques. https://doi.org/10.48550/arXiv.2209.10131*
16. *Malhotra R and Chug A (2016), "Software Maintainability: Systematic Literature Review and Current Trends", International Journal of Software Engineering and Knowledge Engineering, Vol. 26, No. 8, pp1221–1253.*
17. *Şahin C B (2021), "The Role of Vulnerable Software Metrics on Software Maintainability Prediction", European Journal of Science and Technology, Vol. 23, pp686 696. https://doi.org/10.31590/ejosat.858720*

## AUTHORS

**Anita Devi**, Department of Computer Science & Engineering, Om Sterling Global University, NH-52, Hisar-Chandigarh Road, Hisar, Haryana - 125001, India
Email: anita0863@gmail.com

**Saurabh Charaya**, Department of Computer Science & Engineering, Om Sterling Global University, NH-52, Hisar-Chandigarh Road, Hisar, Haryana - 125001, India
Email: saurabh.charaya@gmail.com

**Mukesh Maan**, Department of Computer Science & Engineering, Indian Institute of Information Technology, IIIT Delhi Sonipat Campus, SH 11, Khewra, Sonipat, Haryana – 131 001
Email: mukesh.maan@iiitsonepat.ac.in