Journal of Information Technology Research
Volume 15 • Issue 1

In-Depth Analysis and Prediction of Coupling Metrics of Open Source Software Projects

Munish Saini, Guru Nanak Dev University, Amritsar, India*
https://orcid.org/0000-0003-4129-2591

Raghuvar Arora, Guru Nanak Dev University, Amritsar, India

Sulaimon Oyeniyi Adebayo, Guru Nanak Dev University, Amritsar, India

ABSTRACT

This research was conducted to perform an in-depth analysis of the coupling metrics of 10 open source software (OSS) projects obtained from the Comets dataset. More precisely, the authors analyze the dataset of object-oriented OSS projects (having 17 code-related metrics such as coupling, complexity, and size metrics) to (1) examine the relationships among the coupling and other metrics (size, complexity), (2) analyze the pattern in the growth of software metrics, and (3) propose a model for prediction of coupling. To generalize the model of coupling prediction, they have applied different machine learning algorithms and validated their performance on similar datasets. The results indicated that the random forests algorithm outperforms all other models. The relation analysis specifies the existence of strong positive relationships between the coupling, size, and complexity metrics while the pattern analysis pinpointed the increasing growth trend for coupling. The obtained outcomes will help the developers, project managers, and stakeholders in better understating the state of software health.

KEYWORDS

Coupling, Open Source Software (OSS), Pattern Analysis, Prediction, Software Metrics, Software Quality

1. INTRODUCTION

The measurement of software metrics (Fenton et al., 2002) helps in analyzing and evaluating the Open Source Software (OSS) projects. The development paradigm (Feller & Fitzgerald, 2000) of OSS projects raises many questions when compared with that of proprietary software projects (Tapscott & Caston, 1993). Moreover, in recent times the shift is observed in the software development paradigm from proprietary software to the OSS development paradigm (Tapscott & Caston, 1993). The software quality is one of the major concerns, while the developer proposes or use the OSS development model. Many researchers have performed the fine-grained (Pascarella et al., 2019) as well as coarse-grained (Zendler et al., 2001) analysis to tackle the quality issue of the OSS development. The quality of

DOI: 10.4018/JITR.301267 *Corresponding Author

software determines the strength of software in terms of low defect density (Slaughter et al., 1998) and ultimately it will decide the success or the failure of the OSS project (Lee et al., 2009). The software metrics obtained from the source code measures the different aspects of software development. These metrics help in evaluating the strength and the weakness of the code such as it is observed that as the software evolves the complexity of the software project increases (Daniel et al., 2009).

Quah & Thwin (2003) predicted the software metrics (number of defects and the number of lines changes) for object-oriented software projects. In the present study, we aim to extend the work of Quah & Thwin (2003) by performing the prediction of coupling metrics of object-oriented OSS projects. The coupling is the measure of the inter-module dependency or association (Allen et al., 2001). For quality software development, we expect to have low coupling between the modules. The coupling helps in quantifying internal software quality (Offutt et al., 1993). Sousa et al. (2019) analyzed the evolution behavior of coupling and evaluated the effect of coupling on software reusability and complexity. They observed an increase in the complexity of the system with coupling evolution. In this similar context of working, we extended the work of Sousa et al. (2019) to propose the general model for the prediction of software coupling metrics. We have investigated the Comets dataset[α] using multiple machine learning prediction algorithms and provided a comparison for all these models. The prediction of the coupling allows the developers, project managers, and other stakeholders to better understating the state of software health. Moreover, it will provide information to understand the software evolution behavior. The coupling prediction for the other object-oriented metrics of the Comets dataset intimates the project manager to perform perfective or preventive maintenance activities. This prediction will provide the measure of one dimension of internal software quality. Further, the composite analysis is performed to analyze the pattern existing in the evolution of different object-oriented metrics of OSS projects. The pattern analysis using advanced machine learning algorithms helps in nudging the mute existence of relationship patterns among the software metrics. It will allow analyzing the effect of the coupling metric on other software metrics (size and complexity metrics).

In general, we aim to achieve the following objectives:

1. Examine the relationships among the coupling metrics and other metrics (size and complexity metrics).
2. Analyze the pattern in the growth of different software metrics.
3. Propose a model for the prediction of software coupling metrics (fan-in and fan-out).

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 explains the analysis methodology. Section 4 presents the results and discussion on the outcomes. The last section concludes the paper and provides future directions.

2. RELATED WORK

The quality of the software is extensively dependent on many factors. Modifiability, testability, flexibility, and maintainability are at the forefront of these factors (Guveyi et al., 2020). The user expectations and satisfaction which are key criteria in software productions strongly depend on software quality (Yadav, S., & Kishan, 2020). Most users have their key expectations on the quality of the software than other components (Yue, 2019). Usage of deficient predictors is unceasing research in the software community (Yucalar et al., 2020) and employment of software quality prediction models at the initial stage of the software development lifecycle (SDLC) can help in risk minimization as well as reducing manpower and cost (Jayanthi & Florence, 2019). Broadly, it is considered that the quality can be measured by classifying the metrics into two categories: a) internal quality metrics and b) external quality metrics (Azuma, 1996). The researchers have used

these measures to evaluate software quality from different perspectives. The work of Chidamber et al., (1994) was the pioneer in the field of measuring the object-oriented software metrics. They proposed the suite of six design metrics for object-oriented software projects that will provide insightful information about the internal quality behavior of the software. Later many other researchers (Li & Henry, 1993; Lorenz & Kidd, 1994; Briand et al., 1999, and so on.) have extended the work of Chidamber et al., (1994).

Sousa et al., (2019) investigated the internal quality of the software in facets of the coupling metric. They analyzed the coupling evolution behavior and also measured the effect of coupling metrics on other sets of metrics such as reusability and complexity of the software. They pinpointed the fact that the coupling has a direct impact on the other considered factors. Besides this, we found a gap in the existing study as it is mentioned by Sousa et al., (2019) in their specified future work. They indicated that in the future they would like to propose the global model for the prediction of the coupling metric. More specifically, the prediction of coupling evolution by proposing the general model of prediction. In this present study, we aim to explore this future objective as mentioned in their study. We propose to provide the general model of coupling prediction by using a similar dataset. Furthermore, we looking forward to examining the pattern in the coupling, size, and complexity metrics to uncover the minute relation if it exists between them.

Arisholm et al., (2004) investigated the object-oriented software projects to measure the dynamic coupling and proposed the different dynamic coupling metrics. Further, they analyzed the relationship between the coupling measures and the change proneness of the classes. They specified in the results the presence of relationships among them. In the present study, we aim to study the static coupling measure of 10 object-oriented software projects having 17 software metrics including size, complexity, and coupling metrics.

Quah & Thwin (2003) used the neural networks for estimating the software quality using the object-oriented software metrics. They predicted the number of defects in a class and the number of lines changed per class. Moreover, they specified that the performance of the general regression neural model is better and more accurate than the contrasting ward network model. In this present study, our objectives are some not only in examining the performance of different prediction algorithms but we have the concern to provide a general model for the coupling prediction. Besides this, we aim to find the existing pattern among the considered object-oriented metrics.

Juneja (2019) employed a fuzzy-filtered neuro-fuzzy framework for software fault prediction and was able to generate a framework with high accuracy. Other researchers have also used a different approach to fault prediction analysis. For example, Ha, et al. (2019) used a machine learning model while Al Qasem et al., (2020) used Deep Learning Algorithms. Kanmani et al., (2004) predicted the fault ratio by using the General regression neural network. They performed the specified study on the software developed by the students of their institute. Further, Kanmani et al., (2007) extended their previous work for fault prediction by using a different set of neural networks (Probabilistic and Back prorogation neural networks). They specified that neural networks outperform the statistical models. They also mentioned that the Probabilistic Neural Networks outperforms the back prorogation neural network for the prediction of faults.

The present status of related work in the way to evaluate the object-oriented metrics is more towards finding the applications of machine learning algorithms or to validate the one model with others on different metrics. In this present study, we start the work with the prediction of the coupling metrics of the object-oriented software and further looking in-depth for examining the pattern in the relation of the coupling metric with other size and complexity metrics. The proposed study will provide insight information about the internal quality behaviour that will be determined with the detailed analysis of the coupling metric. It allows the project managers and other stakeholders to better understand and execute timely, the maintenance activities that are required in the smooth run of any software.

3. ANALYSIS METHODOLOGY

3.1 Data Collection

The proposed study is an extension of the work done by Sousa et al., (2019) to measure the quality of software. Sousa et al., (2019) have examined the coupling evolution behaviour and also analyzed the effect of coupling on other sets of metrics (such as reusability and complexity, etc.). For this evaluation, they have used the COMETS (CODE METRIC TIME SERIES) a public dataset. The COMETS dataset consists of a total of 17 source code evolution metrics that will provide insight information about the software internal quality issue in terms of providing size related, complexity related, and coupling related information of 10 Java-based projects (Couto et al., 2013). To perform the present study, we have used a similar dataset but for gaining the different objectives that are to propose the general model of prediction of coupling and to look for the pattern of evolution or relation between different source code metrics (see Table 1). For further analyzes, we have taken the monthly average of all the metrics. Since the processed data is less for applying the prediction algorithms. We have used the bootstrap method for the resampling of the data (Efron & Tibshirani, 1997). The monthly data of software metrics is bootstrapped to 1500 samples by considering the sampling without replacement for the instances of OSS project respectively.

3.2 Software Code Metrics

The COMETS dataset consists of 17 source code metrics that are measured at the level of source code classes. It broadly includes 9 size metrics (Number of attributes (NOA), Number of public attributes (NOPA), Number of private attributes (NOPRA), Number of attributes inherited (NOAI), Number of lines of code (LOC), Number of methods (NOM), Number of public methods (NOPM), Number of private methods (NOPRM), Number of methods inherited (NOMI)), 2 coupling metrics (Fan-in, Fan-out), and 6 CK metrics (Weighted Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number Of Children (NOC), Coupling Between Classes (CBC), Response For a Class (RFC), Lack of Cohesion in Methods (LCOM)) (see Table 2). These sets of metrics provide the information for the evolution of the considered projects (shown in Table 1).

3.3 Random Forests Algorithm

The random forest is an ensemble learning method consists of a combination of tree predictors whereby every tree depends on the values of a random vector sampled independently and with the same

Table 1. Description of COMETS dataset (Couto et al., 2013)

OSS project	Time frame	Versions
Eclipse JDT Core	07/01/2001 - 06/14/2008	183
Eclipse PDE UI	06/01/2001 - 09/06/2008	191
Equinox Framework	01/01/2005 - 06/14/2008	91
Hibernate Core	06/13/2007 - 03/02/2011	98
JabRef	10/14/2003 - 11/11/2011	212
Lucene	01/01/2005 - 10/04/2008	99
Pentaho Console	04/01/2008 - 12/07/2010	72
PMD	06/22/2002 - 12/11/2011	248
Spring Framework	12/17/2003 - 11/25/2009	156
TV-Browser	04/23/2003 - 08/27/2011	221

Table 2. Description of source code metrics

Software Metric	Description
NOA	It includes the total number of attributes in the class.
NOPA	It includes the total number of public attributes in the class.
NOPRA	It specifies the total number of attributes that are private in the class.
NOAI	It indicates the count of the total number of attributes that are inherited in the class.
LOC	It represents the count of the number of lines in the source code.
NOM	It indicates the total number of methods in the class.
NOPM	It indicates the total number of methods that are declared as public in the class.
NOPRM	It indicates the total number of private methods in the class.
NOMI	It specifies the count of the total number of attributes that are inherited in the class.
Fan-in	It indicates the number of modules called this module.
Fan-out	It specifies the count for the number of modules calling the module under examination.
WMC	It represents the measure of complexity in the class.
DIT	It specifies the maximum length from the class to the root class or hierarchy nesting level.
NOC	It represents the number of sub-classes of a class.
CBC	It indicates the number of classes to which the class is coupled.
RFC	Set of methods that are executed in the response of the object of the class under consideration.
LCOM	It specifies the lack of cohesion count for a method in the class.

distribution for all trees in the forest (Breiman, 2001). A brief application of the random forest method is given below. We considered, a dataset D with each metric as a feature and each row representing the corresponding metric value at a period of 2-weeks. A sample is bootstrapped from the dataset and further a subset of features is drawn randomly from the previously obtained sample. Then a decision tree is trained using the random subset from the previous step. This process is repeated *B* number of times to generate a large number of random decision trees. This randomness of features keeps the variance and the bias to the minimum resulting in an effective model for the task.

Random forest regressor

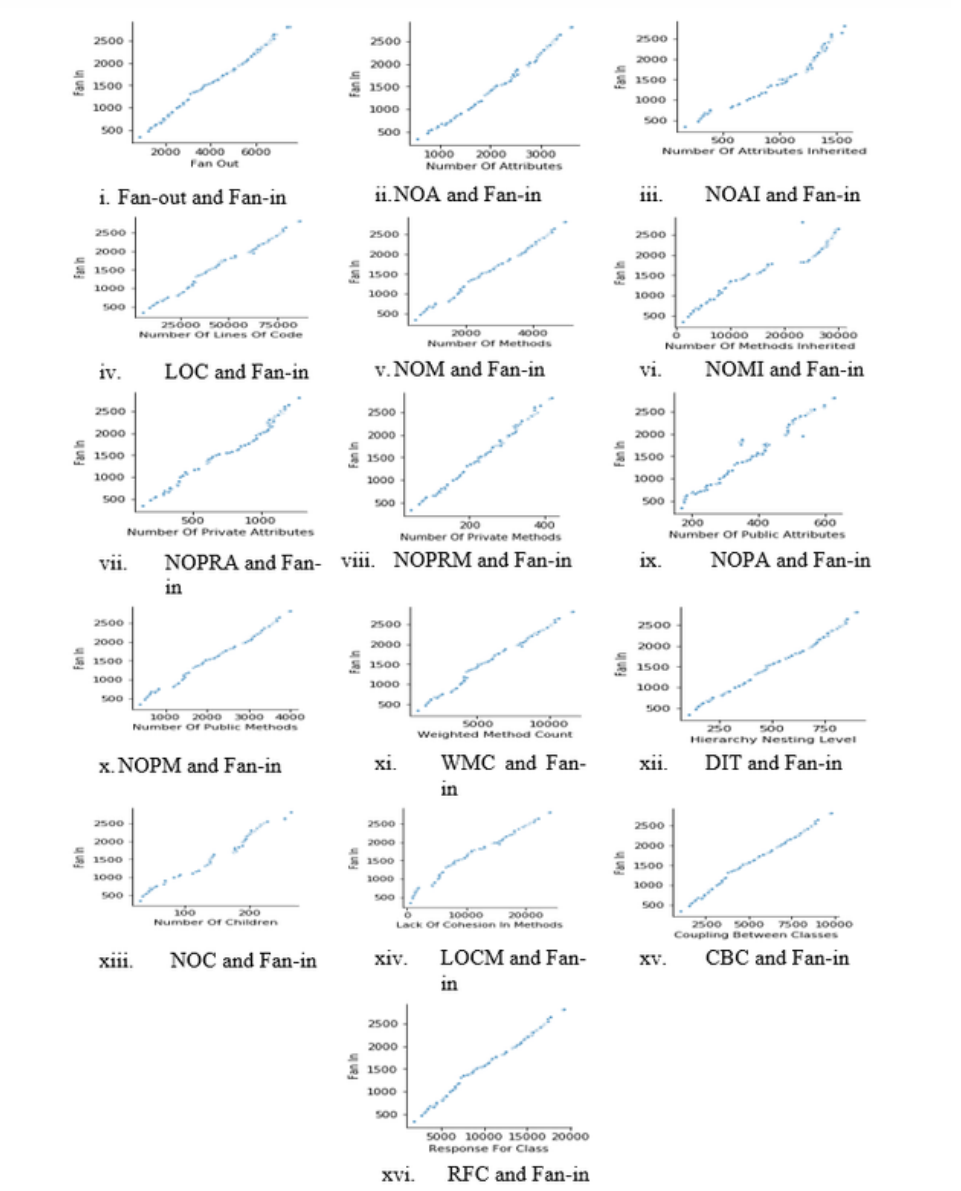
```
Input D: Training Data
Output: Prediction of Coupling
1: procedure RANDOMFORESTCLASSIFIER(D)        ▶D is the labelled
training data
2: forest = new Array ()
3: for i = 0 to B
4: D_i = Bagging(D)                           ▶Bootstrap Aggregation
5: T_i = new DecisionTree ()
6: features_i = RandomFeatureSelection (D_i)
7: T_i .train(D_i , features_i )
8: forest .add(T_i )
9: end for
10: return forest
11: end procedure
```

4. RESULTS AND DISCUSSION

4.1 Analyzing the Relation Among Different Metrics

Figure 1 (i-xvi) summarizes the results of exploratory data analysis on the dataset. Each graph represents a relationship between the two metrics on the axes of it. Every plotted point is the total aggregation of the respective metrics obtained at a regular period (2-weeks). Figure 1-i shows a positive linear relationship between Fan-in and Fan-out. An increase or decrease in Fan-in will cause a proportional increase or decrease in the Fan-out.

Figure 1. Fan-in Metric Correlation against Other Metrics



Similarly, Figure (ii-xvi) presents the relationship of Fan-in metric with other metrics, namely Number of attributes (NOA), Number of attributes inherited (NOAI), Number of lines of code (LOC), Number of methods (NOM), Number of methods inherited (NOMI), Number of private attributes (NOPRA), Number of private methods (NOPRM), Number of public attributes (NOPA), Number of public methods (NOPM), Weighted Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number Of Children (NOC), Lack of Cohesion in Methods (LCOM), Coupling Between Classes (CBC), and Response For a Class (RFC) respectively. It should be noted the combination of all these metrics with Fan-in exhibit similar trends. They increase as the value of Fan-in increases i.e. it depicts the linear relationship of Fan-in with other sets of metrics. The analogous trend was observed after plotting each of these metrics with Fan-out. Having carefully analyzed ten different open source software, we selected three of these systems for our prediction purposes because of their enormous amount of data set. This three software includes; JabRef, PMD, and TV-Browser.

To further establish the aforementioned observations, the Person correlation method was used to calculate the correlation coefficients (Mudelsee, 2003). The Pearson correlation coefficient between two attributes can be calculated using the following equation 1:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - \left(\sum x_i\right)^2}\sqrt{n\sum y_i^2 - \left(\sum y_i\right)^2}} \tag{1}$$

$r_{xy} = Pearson\ r\ correlation\ coefficient\ between\ x\ and\ y$

$n = number\ of\ observations$

$x_i = value\ of\ x\left(for\ i^{th}\ observation\right)$

$y_i = value\ of\ y\left(for\ i^{th}\ observation\right)$

While performing the correlation analysis, it was observed that the correlation coefficients for software metrics of the considered software projects are very high (more than 0.95) which indicates a strong positive correlation between all these metrics.

4.2 Prediction of Fan-in and Fan-out

Coupling metrics were predicted with four selected models; AdaBoost, Random Forests, XGBoost, and KNN Regressor. AdaBoost pools weak algorithms to form a stronger one (Chourey et al., 2019), Random forests are efficient in dealing with lesser sample size and provides a distinctive combination of prediction accuracy (Qi, 2012), XGBoost can be used for both classification and regression (Chen & Guestrin, 2016) while KNN Regressor utilizes local information (Zhou et al., 2005). The models were accessed based on three (3) evaluation metrics; Mean Square Error (MSE), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE).

MSE calculates the sum of the square of the difference between actual and predicted values (refer to Eq. 2). Its value ranges between 0 and 1, where 1 depicts the worst model while 0 value represents the best model:

$N = number\ of\ observations$

$y_i = actual\ output$

$\widehat{y}_i = model\ predicted\ value$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y}_i)^2 \tag{2}$$

MAE evaluates the absolute difference between actual and predicted values (refer to Eq. 3). Moreover, it is not as sensitive to outliers as MSE (Brassington, 2017):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \widehat{y}_i\right| \tag{3}$$

RMSE is the measure of error rate, it is calculated by taking the square root of MSE (see Eq. 4). Both the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) are often employed in model evaluation research (Chai & Draxler, 2014):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y}_i)^2} = \sqrt{MSE} \tag{4}$$

Table 3 presents the values of the mentioned statistical measures for every model. Upon evaluation and thorough comparison of performances of the above given state-of-the-art repressors, it was established that Random Forests outperforms other algorithms, thereby making it an appropriate choice for our research.

Figure 2, presents a chart of relative importance with fifteen (15) different metrics. The total relative feature importance of the four (4) metrics with the highest relative importance; Response For Class, Number of Public Methods, Number of Methods inherited, and Coupling Between Classes is more than 0.85. Beyond the 4th metric, the relative importance drastically decreases to 0.03 and continues to decrease further with the rest of the metrics. We ignore the metrics with the relative importance of less than 0.03 and retrain Random Forests again with the remaining features. The new model was again evaluated based on the four metrics with relative importance greater than 0.03. MSE= 731.23, MAE=22.06 and RMSE=27.041, which gives a negligible difference in the results.

Finally, we predicted values for datasets with the longest time frames, namely; PMD, TV-Browser, JabRef. Each of these 3 datasets covers more than 8 years. The predictions were made for three years with six-month intervals for each dataset. Figure 3-4, presents the actual and predicted values of Fan-in and Fan-out for JabRef software respectively. The predicted and actual values for the second and third months for the Fan-in chart has a difference of 0 each, the third having 1 while others also have very close values. Also, the Fan-out values exhibit a very similar trend of having close values.

Figures 5 and 6 present the actual and predicted values of Fan-in and Fan-out for PWD software respectively. The values for Fan-out has lesser differences than that of Fan-in for the PWD software.

Table 3. Selected Predicting Models

Algorithm	MSE	MAE	RMSE
AdaBoost	4732.05	49.28	68.78
Random Forests	739.68	22.23	27.197
XGBoost	1945.76	33.46	44.11
KNN Regressor	2669.79	34.59	51.67

Figure 2. Relative Importance Chart

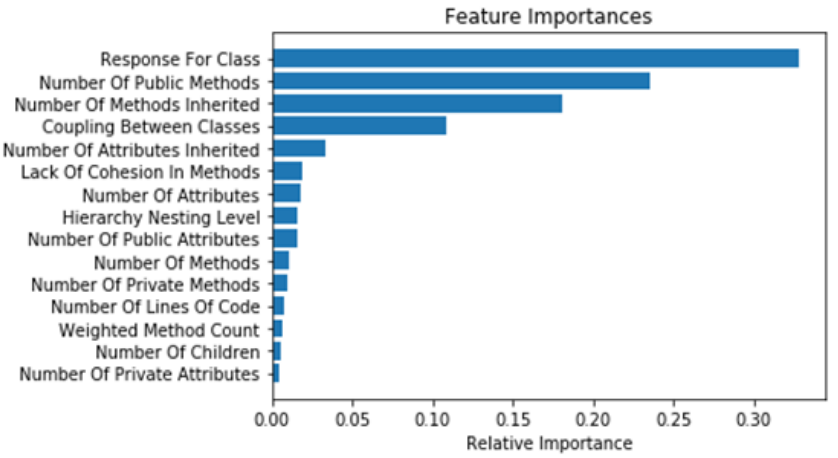
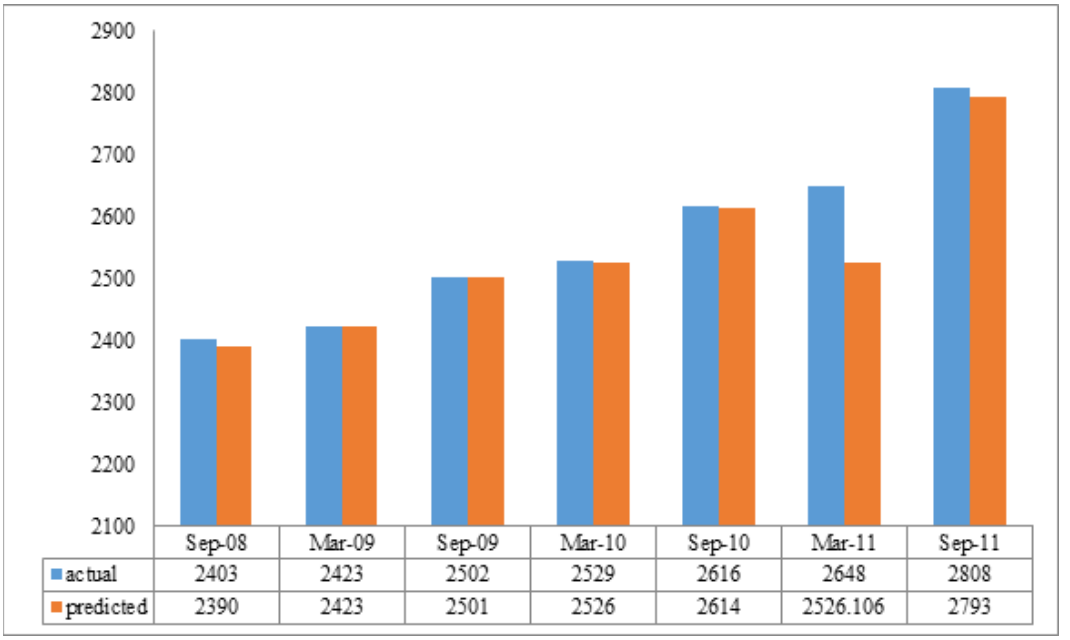


Figure 3. JabRef Fan-in

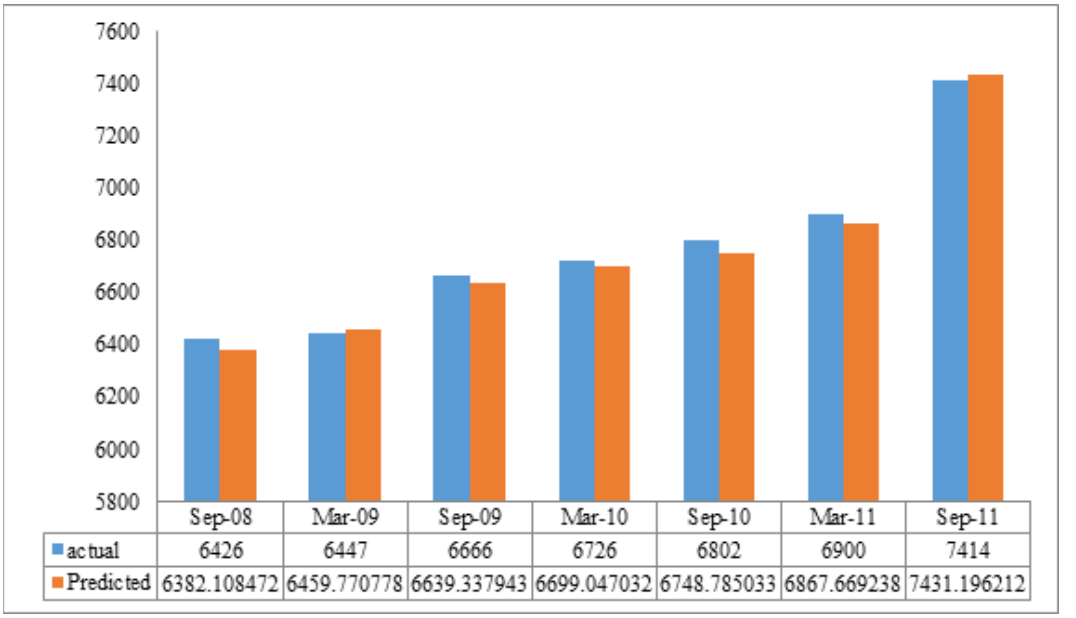


	Sep-08	Mar-09	Sep-09	Mar-10	Sep-10	Mar-11	Sep-11
actual	2403	2423	2502	2529	2616	2648	2808
predicted	2390	2423	2501	2526	2614	2526.106	2793

Furthermore, Figures 7 and 8 depict the actual and predicted values of Fan-in and Fan-out for TVBrowser software respectively. The values for Fan-out and Fan-out also exhibit similar trends like JabRef software.

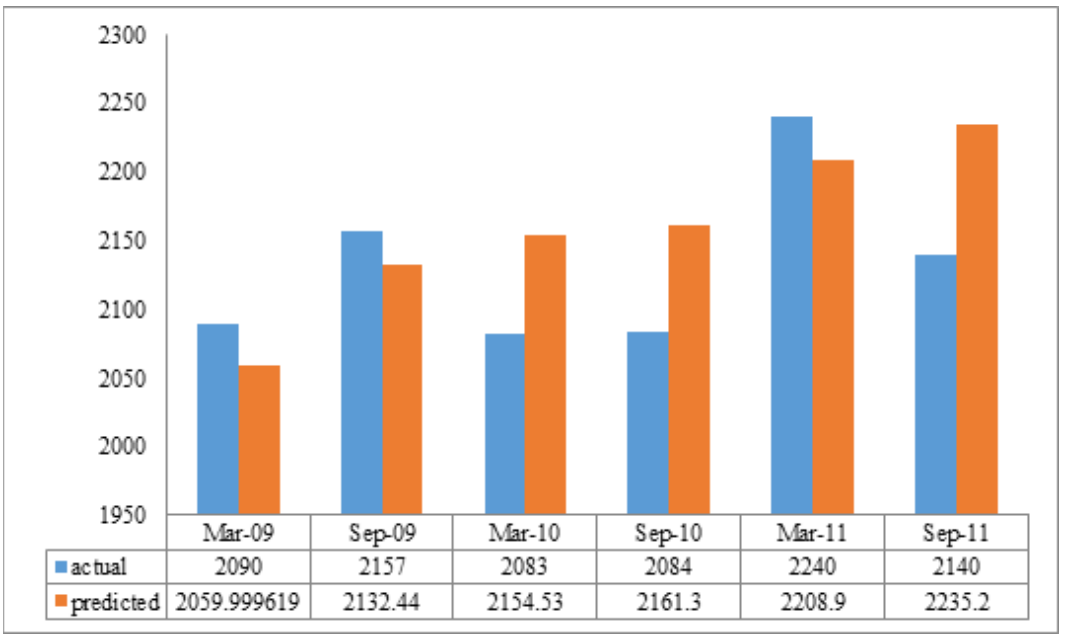
Table 4 specifies the combined Fan-in and Fan-out values for the three software and the accuracies of the model for each dataset are given in Figure 9. From all this analysis, we conclude that random forests predict the values of coupling metrics with higher accuracy in comparison to all other considered methods.

Figure 4. JabRef Fan-out



	Sep-08	Mar-09	Sep-09	Mar-10	Sep-10	Mar-11	Sep-11
actual	6426	6447	6666	6726	6802	6900	7414
Predicted	6382.108472	6459.770778	6639.337943	6699.047032	6748.785033	6867.669238	7431.196212

Figure 5. PWD Fan-in



	Mar-09	Sep-09	Mar-10	Sep-10	Mar-11	Sep-11
actual	2090	2157	2083	2084	2240	2140
predicted	2059.999619	2132.44	2154.53	2161.3	2208.9	2235.2

Figure 6. PWD Fan-out

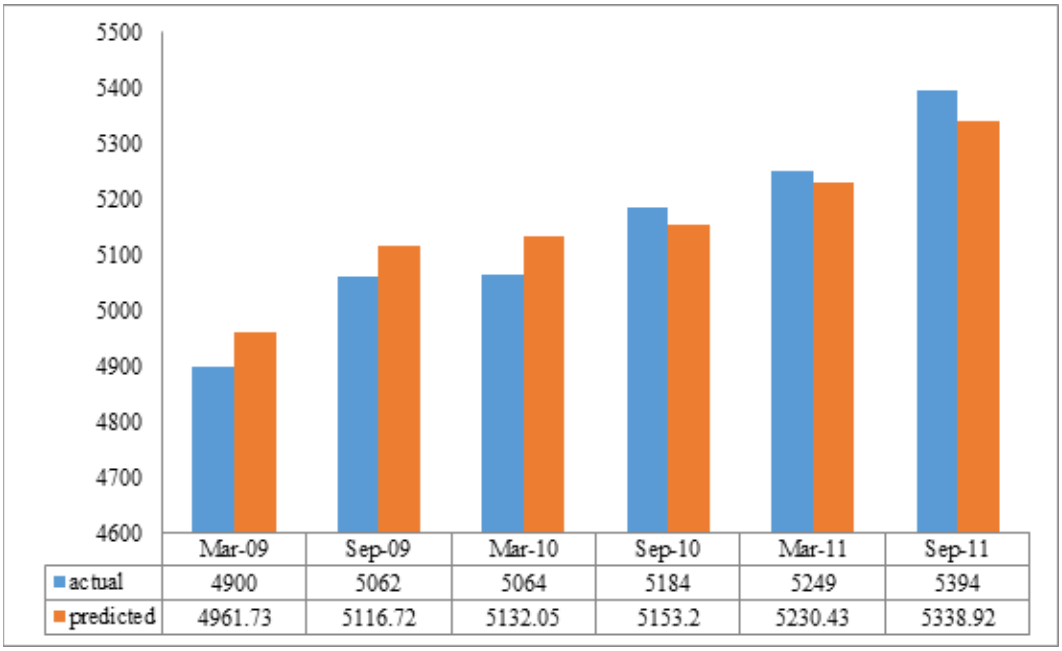


Figure 7. TVBrowser Fan-in

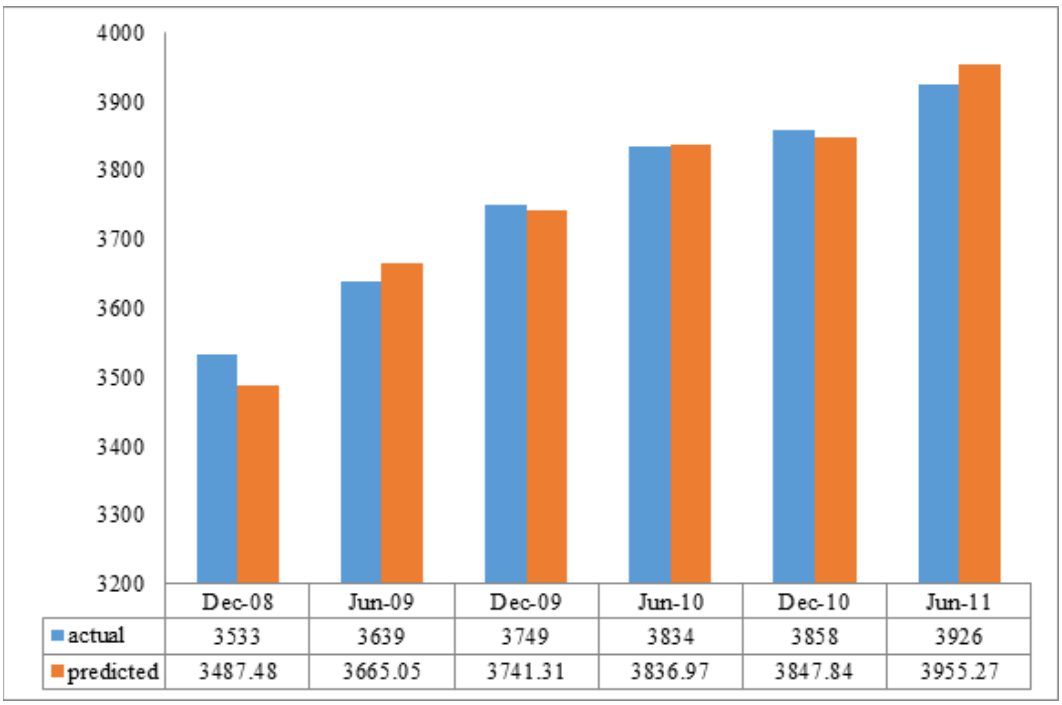


Figure 8. TVBrowser Fan-out

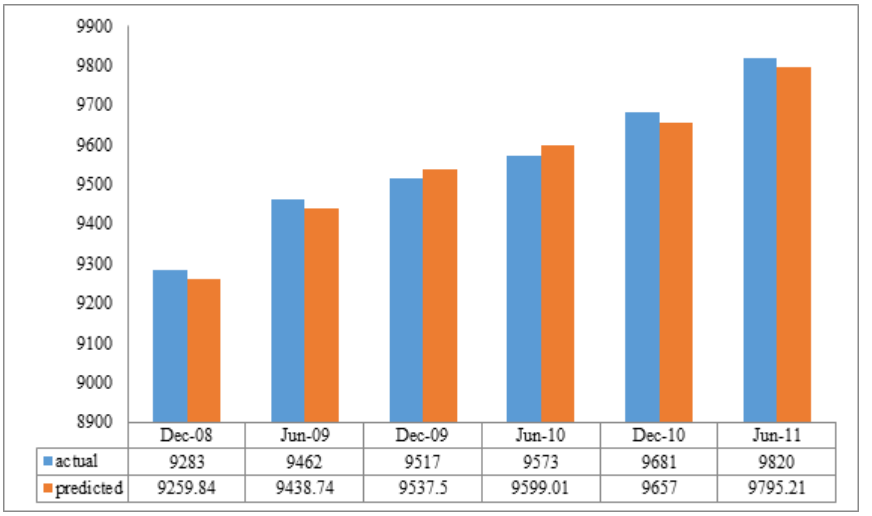
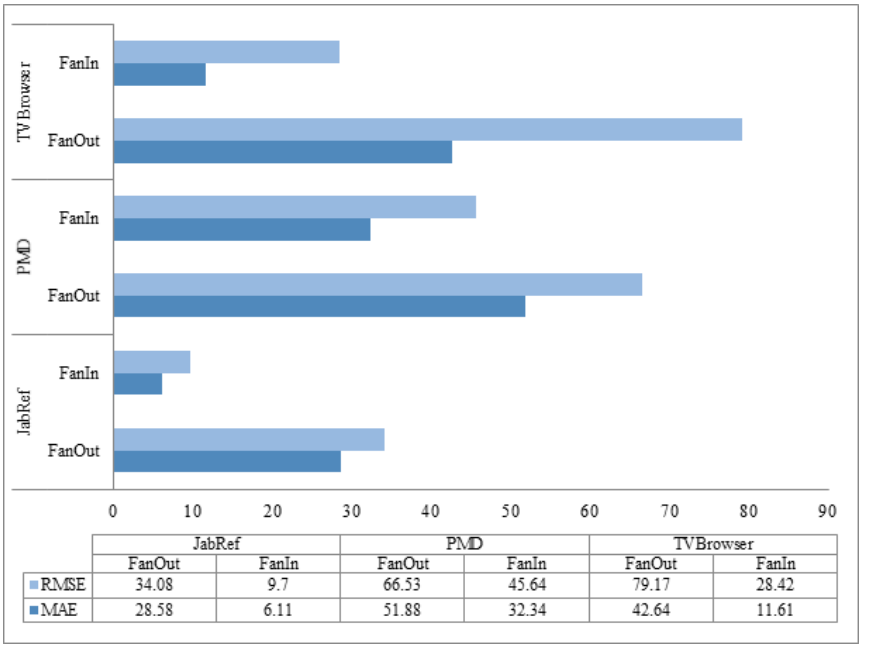


Table 4. Combined Fan-in and Fan-out values

	JabRef		PMD		TVBrowser	
	Fan-out	**Fan-in**	**Fan-out**	**Fan-in**	**Fan-out**	**Fan-on**
MAE	28.58	6.11	51.88	32.34	42.64	11.61
RMSE	34.08	9.7	66.53	45.64	79.17	28.42
MSE	1161.58	94.18	4427.05	2083.24	6268.58	808.14

Figure 9. Accuracies of the model for each dataset



5. CONCLUSION

The quality of the software is the indicator of its reliability, efficiency, maintainability, and security. Most of the time, OSS project development faces the issue of maintaining the good quality of software. The project managers and developers use a different set of methods, techniques, and algorithms to evaluate the quality of the OSS project using an assorted set of metrics. For any quality software development, it is recommended to have low coupling and high cohesion among the modules of the software. In this study, we proceeded to intend to find the relation among the set of code metrics (coupling, size, and complexity metrics) of OSS projects. Furthermore, we applied several machine learning algorithms to predict the coupling metrics (Fan-in and Fan-out). It was observed that random forests outperform in comparison to all other prediction models. The proposed method of analyzing and predicting the coupling metrics from OSS source code will act as an asset for the project managers, software developers, and other stakeholders to predict the quality of software and to look for the pattern of software growth. Moreover, it will provide information to understand the software evolution behavior. In the future, we look forward to repeating and validating a similar experiment on the larger dataset. Furthermore, to better understand and to get a generalized growth pattern of OSS projects, cluster analysis can be performed.

FUNDING AGENCY

The publisher has waived the Open Access Processing fee for this article.

REFERENCES

Al Qasem, O., Akour, M., & Alenezi, M. (2020). The Influence of Deep Learning Algorithms Factors in Software Fault Prediction. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 63945–63960. doi:10.1109/ACCESS.2020.2985290

Allen, E. B., Khoshgoftaar, T. M., & Chen, Y. (2001, April). Measuring coupling and cohesion of software modules: an information-theory approach. In *Proceedings Seventh International Software Metrics Symposium* (pp. 124-134). IEEE.

Arisholm, E., Briand, L. C., & Foyen, A. (2004). Dynamic coupling measurement for object-oriented software. *IEEE Transactions on Software Engineering*, *30*(8), 491–506. doi:10.1109/TSE.2004.41

Azuma, M. (1996). Software products evaluation system: Quality models, metrics and processes—International Standards and Japanese practice. *Information and Software Technology*, *38*(3), 145–154. doi:10.1016/0950-5849(95)01069-6

Brassington, G. (2017). Mean absolute error and root mean square error: which is the better metric for assessing model performance? *EGUGA, 3574*.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. doi:10.1023/A:1010933404324

Briand, L. C., Daly, J. W., & Wust, J. K. (1999). A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering*, *25*(1), 91–121. doi:10.1109/32.748920

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, *7*(3), 1247–1250. doi:10.5194/gmd-7-1247-2014

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). doi:10.1145/2939672.2939785

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object-oriented design. *IEEE Transactions on Software Engineering*, *20*(6), 476–493. doi:10.1109/32.295895

Chourey, A., Phulre, S., & Mishra, S. (2019). Employee attrition prediction using various machine learning techniques. *The International Journal of Analytical and Experimental Modal Analysis, 11*(11), 2718-2724.

Couto, C., Maffort, C., Garcia, R., & Valente, M. T. (2013). COMETS: A dataset for empirical research on software evolution using source code metrics and time series analysis. *Software Engineering Notes*, *38*(1), 1–3. doi:10.1145/2413038.2413047

Daniel, S., Stewart, K., & Darcy, D. (2009). Patterns of evolution in open source projects: A Categorization Schema and Implications. *Patterns of Evolution in Open Source Projects: A Categorization Schema and Implications*.

Efron, B., & Tibshirani, R. (1997). Improvements on cross-validation: The 632+ bootstrap method. *Journal of the American Statistical Association*, *92*(438), 548–560.

Feller, J., & Fitzgerald, B. (2000). A framework analysis of the open source software development paradigm. In *ICIS 2000 proceedings of the twenty first international conference on information systems* (pp. 58-69). Association for Information Systems (AIS).

Fenton, N. E., & Neil, M. (2000, May). Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 357-370). Academic Press.

Guveyi, E., Aktas, M. S., & Kalipsiz, O. (2020, July). Human Factor on Software Quality: A Systematic Literature Review. In *International Conference on Computational Science and Its Applications* (pp. 918-930). Springer. doi:10.1007/978-3-030-58811-3_65

Ha, T. M. P., Tran, D. H., Hanh, L. T. M., & Binh, N. T. (2019, October). Experimental Study on Software Fault Prediction Using Machine Learning Model. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)* (pp. 1-5). IEEE.

Jayanthi, R., & Florence, L. (2019). Software defect prediction techniques using metrics based on neural network classifier. *Cluster Computing*, *22*(1), 77–88. doi:10.1007/s10586-018-1730-1

Juneja, K. (2019). A fuzzy-filtered neuro-fuzzy framework for software fault prediction f or inter-version and inter-project evaluation. *Applied Soft Computing*, *77*, 696–713. doi:10.1016/j.asoc.2019.02.008

Kanmani, S., Uthariaraj, V. R., Sankaranarayanan, V., & Thambidurai, P. (2004). Object oriented software quality prediction using general regression neural networks. *Software Engineering Notes*, *29*(5), 1–6.

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

Lee, S. Y. T., Kim, H. W., & Gupta, S. (2009). Measuring open source software success. *Omega*, *37*(2), 426–438.

Li, W., & Henry, S. (1993). Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, *23*(2), 111–122.

Lorenz, M., & Kidd, J. (1994). *Object-oriented software metrics: a practical guide*. Prentice-Hall, Inc.

Mudelsee, M. (2003). Estimating Pearson's correlation coefficient with bootstrap confidence interval from serially dependent time series. *Mathematical Geology*, *35*(6), 651–665.

Offutt, A. J., Harrold, M. J., & Kolte, P. (1993). A software metric system for module coupling. *Journal of Systems and Software*, *20*(3), 295–308.

Pascarella, L., Palomba, F., & Bacchelli, A. (2019). Fine-grained just-in-time defect prediction. *Journal of Systems and Software*, *150*, 22–36.

Qi, Y. (2012). Random forest for bioinformatics. In *Ensemble machine learning* (pp. 307–323). Springer.

Quah, T. S., & Thwin, M. M. T. (2003, September). Application of neural networks for software quality prediction using object-oriented metrics. In *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings* (pp. 116-125). IEEE.

Slaughter, S. A., Harter, D. E., & Krishnan, M. S. (1998). Evaluating the cost of software quality. *Communications of the ACM*, *41*(8), 67–73.

Sousa, B. L., Bigonha, M. A., & Ferreira, K. A. (2019, September). Analysis of Coupling Evolution on Open Source Systems. In *Proceedings of the XIII Brazilian Symposium on Software Components, Architectures, and Reuse* (pp. 23-32). Academic Press.

Tapscott, D., & Caston, A. (1993). *Paradigm shift: The new promise of information technology* (Vol. 15). McGraw-Hill.

Yadav, S., & Kishan, B. (2020). Analysis and Assessment of Existing Software Quality Models to Predict the Reliability of Component-Based Software. *International Journal of Emerging Trends in Engineering Research, 8*(6).

Yucalar, F., Ozcift, A., Borandag, E., & Kilinc, D. (2020). Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability. *Engineering Science and Technology, an International Journal, 23*(4), 938-950.

Yue, C. (2019). A projection-based approach to software quality evaluation from the users' perspectives. *International Journal of Machine Learning and Cybernetics*, *10*(9), 2341–2353.

Zendler, A., Pfeiffer, T., Eicks, M., & Lehner, F. (2001). Experimental comparison of coarse-grained concepts in UML, OML, and TOS. *Journal of Systems and Software*, *57*(1), 21–30.

Zhou, Z. H., & Li, M. (2005). Semi-Supervised Regression with Co-Training. IJCAI, 5.

ENDNOTE

[α] http://java.llp.dcc.ufmg.br/comets/

Munish Saini is an Assistant Professor in the Department of Computer Engineering and Technology, Guru Nanak Dev University, Amritsar, India. He received his Ph.D. in Computer Science and Engineering from Guru Nanak Dev University, Amritsar, India. His research interest are in Open Source Software, Software Engineering, Data Mining, and Machine Learning.

Raghuvar Arora is an undergraduate student in the Department of Computer Engineering and Technology, Guru Nanak Dev University, Amritsar, India. His research interest is in Machine Learning.

Sulaimon Oyeniyi Adebayo is a research student in the Department of Computer Engineering and Technology, Guru Nanak Dev University, Amritsar, India. His research interests are in Software Engineering, Social Media Analytics, Text Analytics, and Data Mining.