applied sciences

MDPI

Article

Predicting Software Cohesion Metrics with Machine Learning Techniques

Elif Nur Haner Kırğıl and Tülin Erçelebi Ayyıldız *

Computer Engineering, Baskent University, Ankara 06790, Turkey
* Correspondence: ercelebi@baskent.edu.tr

Abstract: The cohesion value is one of the important factors used to evaluate software maintainability. However, measuring the cohesion value is a relatively difficult issue when tracing the source code manually. Although there are many static code analysis tools, not every tool measures every metric. The user should apply different tools for different metrics. In this study, besides the use of these tools, we predicted the cohesion values (LCOM2, TCC, LCC, and LSCC) with machine learning techniques (KNN, REPTree, multi-layer perceptron, linear regression (LR), support vector machine, and random forest (RF)) to solve them alternatively. We created two datasets utilizing two different open-source software projects. According to the obtained results, for the LCOM2 and TCC metrics, the KNN algorithm provided the best results, and for LCC and LSCC metrics, the REPTree algorithm was the best. However, out of all the metrics, RF, REPTree, and KNN had close performances with each other, and therefore any of the RF, REPTree, and KNN techniques can be used for software cohesion metric prediction.

Keywords: software quality; CK metrics; software cohesion metrics; machine learning; LCOM

check for updates

Citation: Haner Kırğıl, E.N.; Erçelebi Ayyıldız, T. Predicting Software Cohesion Metrics with Machine Learning Techniques. *Appl. Sci.* **2023**, *13*, 3722. https://doi.org/10.3390/app13063722

Academic Editor: Juan Pavón

Received: 30 January 2023
Revised: 1 March 2023
Accepted: 13 March 2023
Published: 15 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Creating high-quality software is one of the main goals of software engineers. The six features that must be provided for a software to be of high quality are explained as follows according to the ISO 9126 standard: functionality, reliability, usability, efficiency, maintainability, and portability [1]. When the quality of the software is improved, the error/fault rate decreases and maintainability increases, and thus it is very important in terms of cost and time in the software life cycle. There are many metrics in the literature to measure the quality of the software. The most accepted metric set to measure software quality in the literature is the Chidamber and Kemerer (CK) metric set [2]. The CK metric set consists of six metrics. These are weighted methods per class (WMC), depth of inheritance tree (DIT), number of children (NOC), coupling between object classes (CBO), response for a class (RFC), and lack of cohesion in methods (LCOM) [2].

Maintenance is one of the most important and expensive phases of the software development lifecycle. Therefore, it is an advantage of quality software to keep the maintenance cost low by attaching importance to the software quality during the development phase [3,4]. Cohesion is also one of the important quality factors that affects maintainability directly, and it can be defined as the degree to which components in a class (an extensible program code template for creating objects) belong to each other [5–7]. Highly cohesive modules are easier to maintain, and a high level of cohesion in a class means that there is only one responsibility in the class, and it is not divided into smaller units.

Since cohesion value is a very important issue for software maintainability, we focused on cohesion metrics in this study. There are many metrics in the literature to measure cohesion, and each of them measures the cohesion of a class with different calculation methods and formulas. It is stated that there are 95 cohesion metrics in the literature [8]. The most widely used and known cohesion metrics in the literature are LCOM1 [9], LCOM2 [10],

LCOM3 [11], LCOM4 [12], LCOM5 [13], tight class cohesion (TCC), and loose class cohesion (LCC) [14]. The most used cohesion measurement metric is LCOM2 [8].

In this study, LCOM2, TCC, LCC, and LSCC metric values were predicted by only using the number of methods (a procedure associated with a message and an object) and attributes. In order to calculate the cohesion value of a class, almost all of the metrics used in the literature considered the methods and attributes that form the basis of the class and their relationship with each other. The extent to which the variables created in a class are used by the methods created in that class directly affects the cohesion value of that class. In the cohesion metric formulas used, the number of methods and the number of attributes is necessarily common. They differ only in determining which situations the relationship between methods and attributes, which are the basic building block of the class, exist and in which situations they do not. Therefore, in this study, the number of methods and the number of attributes are used as independent variables, where these variables are common in other cohesion metrics and are the two main factors in determining the cohesion value of the class.

While selecting the cohesion metrics, attention was paid to the different types of calculation features. All the selected cohesion metrics calculated the cohesion by considering different criteria and using different measurement techniques. The metrics used in this study were selected on the basis of the following properties:

- LCOM2 is the most used cohesion metric;
- TCC metric uses the graph structure and is normalized;
- LCC considers not only direct connections, but also indirect connections;
- LSCC metric not only measures class cohesion but also measures the degree of cohesion.

The dataset was not obtained ready-made, and it was produced within the scope of the study. LCOM2, TCC, LCC, and LSCC metric values of the classes were obtained with the help of the tool, and a dataset was created. By using machine learning methods, the values of the cohesion metrics were estimated with the least error.

To the best of our knowledge, there is no study in the literature that predicts cohesion metrics using machine learning techniques.

2. Related Work

Many metrics and tools have been developed in the literature to measure the cohesion of the software. According to the measured cohesion value, it is possible to comment on the quality of the software. In this section, prominent studies to measure the cohesion of the software are examined.

LCOM1 [9] was described by Chidamber and Kemerer in 1991. It is calculated as the number of method pairs that do not share attributes between. The LCOM1 metric is not normalized between 0 and 1. For LCOM1, the minimum value is 0 and the maximum value is the number of method pairs in a class. Since the LCOM metric measures incompatibility, a low LCOM value means a high cohesion.

LCOM2 [10] was described by Chidamber and Kemerer in 1994. Similar to LCOM1, the LCOM2 metric is not normalized, and a low LCOM2 value indicates a high cohesion. Two parameters are calculated for the calculation of the LCOM2 metric. P is calculated as the number of method pairs that do not share attributes, and Q is calculated as the number of method pairs that share attributes among them. P-Q gives the value of LCOM2.

LCOM3 [11] was proposed by Hitz and Montazeri in 1995. Unlike LCOM1 and LCOM2, the LCOM3 metric provides the calculation using the graphical structure. In the graph created, the nodes show the methods, while the links show the relationship between the methods. If two methods share attributes between them, in other words, if they use same attribute, a link is drawn between those two methods. When the connected nodes are counted as one, the number of independent components provides the value of LCOM3.

The LCOM4 [12] metric was also proposed by Hitz and Montazeri in 1995. It uses a similar calculation method as LCOM3. The only difference with LCOM3 is that in LCOM3, only the relationship of methods is checked according to whether they share attributes

between them, while in LCOM4, in addition to this, the case of methods calling each other is also checked.

The LCOM5 [13] metric was created by Henderson and Sellors in 1996. The LCOM5 metric can take values between 0 and 1. In order to calculate the LCOM5 metric, they utilized the number of attributes, the number of methods, and the sum of the number of attributes used by each method.

TCC and LCC [14] metrics were developed by Bieman and Kang in 1995. A, B, and C are the methods of the class. When method A calls method B, A and B are directly linked. When method A calls method B and method B calls method C, A and C are indirectly related. TCC and LCC metrics also use graph structures such as LCOM3 and LOM4. While TCC only considers direct connections, LCC also considers indirect connections.

LSCC [6] was proposed by Dallal and Briand in 2012. The LSCC metric aims not only to measure whether there is cohesion between the two methods, but also to measure the degree of cohesion, if there is any. The LSCC metric is normalized between 0 and 1. The closer the value is to 1, the better the cohesion. In addition, according to Dallal and Briand's experimental study, LSCC was shown to be one of the top three cohesion metrics in identifying fault occurrences. LSCC is also a normalized cohesion metric between 0 and 1.

In the study conducted by Alzahrani et al. [15] in 2019, the client-based cohesion metric (CCC) was proposed, and it was emphasized that this metric is a powerful factor in predicting maintainability. It was thought that cohesion metrics should be obtained at the design stage, before the implementation stage. It is thought that it cannot fully reflect the class cohesion since the inter-method and method–attribute connections are not yet clear at the design stage.

The modified approach on the LCOM (MALCOM) metric [16] was proposed in the study by Ganesh and Raj. This metric categorizes the cohesion as low, medium, and high according to the result. The difference with the LCOM metric is that while LCOM only checks for the presence of cohesion, MALCOM also measures the level of cohesion. In this case, a numerical value was unable to be obtained by limiting cohesion with only three scales.

The SCCM (scoped class cohesion metric) [17] was proposed in the study by Wanjiku et al., and a tool was developed to calculate this metric. The proposed metric is normalized between 0 and 1. Calculation is provided using numbers of public, private, and protected methods, as well as attributes. With the total value of general and specific cohesion, overall class cohesion was calculated. It is thought that calculating the numbers of private and protected methods and attributes separately and combining them later requires more formulas and effort.

Chen et al. [18] recommended the OntRECoh (Online Ontological Cohesion Assessment for SRE Design Quality) tool in their study. This approach is designed to calculate the cohesion values of the system and to offer improvement suggestions with a web-based interface. In this study, with reverse engineering after the coding, the required design was created with the UML diagram. While it is possible to calculate the source-code-based cohesion, it is thought that it will require extra time and effort to create the redesign diagram over the code created by reverse engineering and then calculate the cohesion.

Many studies have been carried out in the literature to measure the cohesion value of the software. Considering the studies, it is seen that software cohesion has been attempted to be measured with new formulas or tools. Not every tool measures every metric, and thus the user should look up which tool measures which metric, which is a time-consuming process. The aim of our study was to propose a machine-learning-based methodology for predicting cohesion metrics so that our methodology can be easily applied by the researchers. To the best of our knowledge, there is no study in the literature that predicts cohesion metrics using machine learning techniques, and thus our study is the first in this manner.

3. Materials and Methods

In this study, LCOM2, TCC, LCC, and LSCC metrics were predicted to measure software cohesion using machine learning techniques. The reasons for choosing these metrics are explained in detail in the Introduction section. The aim of the study was to predict the cohesion value of the class by using the number of methods and the number of attributes in the class.

3.1. Dataset

Finding the dataset and pre-processing the data are very important processes. The datasets obtained publicly may not have the attributes that are exactly suitable for the purpose of the study. Moreover, there may be different features in the dataset, although it is out of the scope of the study. Removing the unnecessary features directly from the dataset may also cause differences in the result obtained from the analysis. This will reduce the power of the established model. In our study, we created two datasets utilizing the open source Freeplane [19] and Eclipse Plug-in Development Environment (PDE) user interface (UI) software systems (https://github.com/dspinellis/awesome-msr, accessed on 23 February 2023).

To create Dataset#1, the open source Freeplane application's classes were used. Freeplane was obtained from http://sourceforge.net (accessed on 1 December 2022) and developed with Java programming language. Freeplane is a popular open-source software that is downloaded by thousands of people every week and updated frequently by the developers. Moreover, according to the hundreds of reviews, it has a high score. These properties make Freeplane a good representative software to be used in our study. There are 1471 classes in the Freeplane application. It is a medium-sized project consisting of 198.486 lines of code. With the cohesion metrics found in the literature, calculations are generally made on a class basis. Not all the 1471 classes were used in the study. A total of 282 classes were excluded because 170 classes only had the interface definition, 24 classes only had enum definitions, and the content of 6 classes were empty or they only had attribute definitions. In addition, 82 classes were also excluded because they had less than 2 methods, and therefore the cohesion was unable to be calculated.

To create Dataset#2, the open-source Eclipse Plug-in Development Environment (PDE) user interface (UI) classes were used. Eclipse PDE UI was obtained from https://github.com/dspinellis/awesome-msr (accessed on 23 February 2023) and developed with Java like Dataset#1.

There are 1498 classes in the Eclipse PDE UI. A total of 261 classes were excluded because some of them had less than 2 methods, and for some of them, source code could not be reached.

The actual values of the LCOM2, TCC, LCC, and LSCC metrics for the remaining 1189 classes of Dataset#1 and 1237 classes of Dataset#2 were calculated using the software developed by Dallal [6]. The elements of the dataset are the number of methods in the class; the number of attributes; and LCOM2, TCC, LCC, and LSCC metric values. The software used was created with the Java programming language. A total of 16 different cohesion metrics can be calculated by using this software. We should note that this software does not have a user interface and also requires high effort to calculate the metrics since the metrics are obtained by applying each class one by one. Considering that this calculation is made for several hundreds of classes, this is a time-consuming process that takes too long.

3.2. Applied Method

In this study, we tried to make the class cohesion value, which is one of the software quality criteria, to be the closest estimation to the actual value. To do this, we attempted to predict LCOM2, TCC, LCC, and LSCC metrics by using machine learning techniques. KNN [20], REPTree [21], random forest (RF) [22] (100 trees), MLP [23], SVM [24] (polynomial kernel), and linear regression (LR) [25] techniques were used in this study.

Regression analysis is used to make numerical predictions. In regression analysis, the dependent variable is estimated according to the independent variable [26]. In this study, LCOM2, TCC, LCC, and LSCC metric values were estimated by looking at the number of methods and the number of attributes in the class, so the number of methods and the number of attributes was the independent variable, and LCOM2, TCC, LCC, and LSCC were the dependent variables.

Weka 3.8.6 was used for regression analysis in the study. It is an open-source software developed by Weka University of Waikato [27]. It is frequently used in data mining and statistics [28]. Weka is considered a milestone in the use of data mining and machine learning techniques. It is widely accepted in academic and business fields in data mining. Thanks to its open-source code, the development of Weka has also accelerated. It includes various algorithms for regression, classification, pre-processing, data visualization, and clustering. The designed interface also provides easy access to the menus. Therefore, it was preferred that we used the Weka tool in the study. Weka accepts arff extension as file type [29]. Separate arff files were created for LCOM2, normalized LCOM2, TCC, LCC, and LSCC. Each arff file has the number of methods, the number of attributes, and the metric value. These arff files were uploaded to Weka sequentially, and machine learning techniques were applied for each of them in turn.

3.3. Performance Evaluation Metrics

Correlation coefficient (R), mean absolute error (MAE) and root mean squared error (RMSE), which are among the most frequently used metrics, were used to evaluate the results of the regression analysis using machine learning techniques and the established model [30,31].

4. Results and Discussion

In this study KNN, RF, REPTree, LR, MLP, and SVM machine learning techniques were used for predicting software cohesion values. The results of the regression analysis were evaluated over the results of these techniques. In the study, the 10-fold cross-validation technique was used in all the models established. The training set given with this technique was divided into 10 equal parts. One in each cycle was used for testing, and the remaining nine were used for training. When the other cycle was passed, another part was used for testing, and the remaining nine parts were used for training. In this way, the same method was run 10 times using 10 different training and test sets. Since each split piece was used in both training and testing phases, the errors caused by the splits were minimized [32]. The analysis results obtained for the LCOM2 metric are given in Table 1.

Table 1. Regression analysis results of LCOM2 metric.

	Dataset#1			**Dataset#2**		
	R	**MAE**	**RMSE**	**R**	**MAE**	**RMSE**
REPTree	0.937	23.067	105.718	0.947	18.675	53.308
RF	0.973	18.938	71.253	0.968	16,385	41.561
KNN	0.956	21.120	106.143	0.962	16,909	50.282
LR	0.811	90.992	172.806	0.849	50,618	87.800
MLP	0.982	23.813	56.555	0.955	32,170	51.887
SVM	0.823	40.737	240.276	0.857	32.391	112.570

According to Table 1, it is seen that the lowest MAE value was obtained with the random forest algorithm. For the KNN algorithm, different k values were tried, starting from 1. As the k value was changed, the k value continued to be increased over odd numbers if the increase in the correlation coefficient and the decrease in the error rate continued. When the decrease started, the experiment was completed. For the LCOM2

metric, the highest correlation coefficient and the lowest error value were obtained with the KNN algorithm at k = 3. The LCOM2 metric for Dataset#1 took values between 0 and 5749, and for Dataset#2, the LCOM metric took values between 0 and 2393. Therefore, outlier and extreme value analysis were performed on LCOM2 values. Equation (1) was used for outlier analysis and Equation (2) was used for extreme value analysis [33]. In the following formulas, Q3 is the third quartile, IQR is the interquartile range, EVF is the extreme values factor, and OF is the outlier factor.

$$Q3 + OF \times IQR < x \leq Q3 + EVF \times IQR \quad (1)$$

$$x > Q3 + EVF \times IQR \text{ and } x < Q1 - EVF \times IQR \quad (2)$$

As a result of the extreme and outlier analysis, values in the range of $65 < x < 120$ were determined as outliers and $x > 120$ were determined as extreme values for Dataset#1. For Dataset#2, the outlier range was $194 < x < 338$, and extreme values were $x > 338$. The data for LCOM2's actual value which was in these ranges was extracted from the dataset. For Dataset#1, by discarding a total of 137 data, the study was repeated with the remaining 1052 data points. Similarly, for Dataset#2, 90 data points were discarded, and the study was repeated for 1147 data points. After extracting the data as a result of the outlier and extreme value analysis, the remaining data were retrained with the same machine learning techniques. The analysis results obtained after removing the outliers and extreme values are provided in Table 2.

Table 2. Regression analysis results for the LCOM2 metric after outliers and extreme values were removed.

	Dataset#1			**Dataset#2**		
	R	**MAE**	**RMSE**	**R**	**MAE**	**RMSE**
REPTree	0.681	5.245	9.677	0.892	9.348	17.259
RF	0.665	5.272	10.057	0.873	9.584	18.813
KNN	0.709	5.080	9.304	0.895	9.398	17.025
LR	0.607	5.989	10.470	0.861	12.662	19.440
MLP	0.603	6.841	10.653	0.868	12.346	19.107
SVM	0.601	5.886	10.542	0.860	12.125	20.182

According to Table 2, the highest correlation and lowest error values were obtained with the KNN algorithm. The best results for KNN were obtained at k = 9. After removing the extreme values and outliers, the error rates decreased. After removing the extreme values, the same experiment was repeated by normalizing the LCOM2 metric between 0 and 1, and the result is shown in Table 3.

Table 3. Regression analysis results for the normalized LCOM2 metric after removing outliers and extreme values.

	Dataset#1			**Dataset#2**		
	R	**MAE**	**RMSE**	**R**	**MAE**	**RMSE**
REPTree	0.681	0.082	0.151	0.892	0.048	0.090
RF	0.665	0.082	0.157	0.873	0.049	0.098
KNN	0.709	0.079	0.145	0.895	0.048	0.088
LR	0.607	0.094	0.164	0.861	0.065	0.101
MLP	0.603	0.107	0.167	0.868	0.064	0.099
SVM	0.601	0.092	0.165	0.860	0.063	0.105

For the normalized LCOM2 metric, the highest correlation and lowest error values were obtained with the KNN algorithm. The best values for the KNN algorithm were obtained at k = 9. The correlation coefficients were the same as the values given in Table 2. Compared with the data shown in Tables 1 and 2, the reason for the large reduction in error rates was that the LCOM2 value was normalized between 0 and 1. For the remaining metrics, there was no need to apply normalization since these metrics were already between 0 and 1.

The regression analysis results obtained for the TCC metric are provided in Table 4. The highest correlation coefficient and the lowest error values were obtained with the KNN algorithm at k = 13.

Table 4. Regression analysis results for the TCC metric.

	Dataset#1			Dataset#2		
	R	**MAE**	**RMSE**	**R**	**MAE**	**RMSE**
REPTree	0.632	0.234	0.319	0.579	0.201	0.280
RF	0.601	0.238	0.332	0.543	0.209	0.292
KNN	0.640	0.231	0.317	0.586	0.203	0.278
LR	0.217	0.362	0.401	0.147	0.282	0.340
MLP	0.279	0.354	0.409	0.160	0.293	0.352
SVM	0.208	0.361	0.409	0.117	0.267	0.382

According to the Table 5 regression analysis results obtained for the LCC metric. The highest correlation and lowest error values were obtained with the REPTree algorithm. The highest correlation and lowest error values for the KNN algorithm were obtained at k = 5.

Table 5. Regression analysis results for the LCC metric.

	Dataset#1			Dataset#2		
	R	**MAE**	**RMSE**	**R**	**MAE**	**RMSE**
REPTree	0.603	0.259	0.342	0.595	0.230	0.308
RF	0.562	0.266	0.359	0.561	0.235	0.321
KNN	0.584	0.262	0.350	0.582	0.232	0.313
LR	0.239	0.383	0.417	0.224	0.319	0.373
MLP	0.247	0.374	0.427	0.380	0.298	0.372
SVM	0.234	0.381	0.421	0.234	0.298	0.429

When the data in Table 6 for the LSCC metric were examined, the highest correlation and lowest error values were obtained with the REPTree technique. The highest correlation and lowest error values for the KNN algorithm were obtained at k = 5.

Table 6. Regression analysis results for the LSCC metric.

	Dataset#1			Dataset#2		
	R	**MAE**	**RMSE**	**R**	**MAE**	**RMSE**
REPTree	0.762	0.149	0.232	0.896	0.097	0.178
RF	0.757	0.151	0.235	0.894	0.097	0.179
KNN	0.753	0.152	0.236	0.887	0.100	0.185
LR	0.382	0.270	0.331	0.487	0.303	0.351
MLP	0.631	0.223	0.294	0.748	0.204	0.278
SVM	0.358	0.248	0.369	0.484	0.274	0.403

5. Threats to Validity

All empirical studies were subject to threats to validity. Here, we point out the most relevant ones with our case and discuss these threats. Generalization of the methodology proposed in this study is one of the validity threats that should be considered. We note that there was great commonality between the classes utilized in this study—they all belonged to the same project. Another concern was the reliability of the collected data. The data utilized in this study were taken from an open-source project. Hence, they were used and controlled by a large number of developers from all over the world, and thus the data utilized in this study can safely be regarded as reliable.

6. Conclusions

In the study, the cohesion value, which is one of the most important criteria for evaluating software quality, was predicted by RF, KNN, REPTree, SVM, MLP, and LR machine learning techniques. The dataset was produced using the tool during the study process. It was created by using the number of methods; the number of attributes; and the LCOM2, TCC, LCC, and LSCC metric values belonging to 1189 classes of Dataset#1 and 1237 classes of Dataset#2. There are many tools available and used in the literature to obtain software cohesion value. These tools calculate the cohesion value in different ways. Class design should be changed according to the obtained cohesion value. In general, the methods in the class and the structural relations of the attributes with each other are considered to be the measurement techniques. In this study, the class cohesion value was predicted quickly, easily, and practically. The experiment was based on the number of methods and the number of attributes in the class. According to these two values, the cohesion value of the class was estimated. As the cohesion value, LCOM2, TCC, LCC, and LSCC metrics, which measure in different ways and have different properties, were chosen within the scope of the study. To the best of our knowledge, there is no study in the literature that predicts the cohesion metric with regression analysis by using machine learning techniques.

The obtained results differed according to the cohesion criterion used. The LCOM2 metric is not a normalized metric. Outlier, extreme value analysis, and normalization were performed on both Dataset#1 and Dataset#2. All the results obtained were provided separately. According to the obtained results, for the LCOM2 and TCC metrics, the KNN algorithm provided the best results, and for LCC and LSCC metrics, the REPTree algorithm provided the best results. However, for all metrics, RF, REPTree, and KNN had close performances to each other, and therefore any of the RF, REPTree, and KNN techniques can be used for software cohesion metrics prediction.

The study was limited to a total of 1189 (Dataset#1) + 1237 (Dataset#2) classes for TCC, LCC, and LSCC metrics, and 1052 (Dataset#1) + 1147 (Dataset#2) classes remaining after outlier and extreme value analysis in the LCOM2 metric. We aim to increase the class number in future work and to increase the closeness of the estimated value obtained to the actual value by including classes belonging different domains. In addition, the same study can be conducted by choosing different cohesion metrics.

Author Contributions: Conceptualization, E.N.H.K. and T.E.A.; methodology, T.E.A.; software, E.N.H.K.; formal analysis, E.N.H.K.; investigation, E.N.H.K. and T.E.A.; resources, E.N.H.K. and T.E.A.; data curation, E.N.H.K.; writing—original draft preparation, E.N.H.K. and T.E.A.; writing—review and editing, T.E.A.; visualization, E.N.H.K. and T.E.A.; supervision, T.E.A.; project administration, T.E.A.; funding acquisition, not applicable. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author without undue reservation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Obisat, A.; Alhalhouli, F.M.; Alshabatat, T.I. Review of Literature on Software Quality. *World Comput. Sci. Inf. Technol. J.* **2018**, *8*, 32–42.
2. Tiwari, S.; Rathore, S.S. Coupling and Cohesion Metrics for Object-Oriented Software: A Systematic Mapping Study. In Proceedings of the 11th Innovations in Software Engineering Conference, Hyderabad, India, 9–11 February 2018.
3. Tarwani, S.; Chug, A. Assessment of Optimum Refactoring Sequence to Improve the Software Quality of Object-Oriented Software. *J. Inf. Optimiz. Sci.* **2020**, *41*, 1433–1442. [CrossRef]
4. Tarwani, S.; Chug, A. Agile Methodologies in Software Maintenance: A Systematic Review. *Inform.-J. Comput. Inform.* **2016**, *40*, 415–426.
5. Dallal, J.A. Improving Object-Oriented Lack-of Cohesion Metric by Excluding Special Methods. In Proceedings of the 10th WSEAS International Conference on Software Engineering Parallel and Distributed Systems, Cambridge, UK, 20–22 February 2011; pp. 124–129.
6. Dallal, A.; Briand, L.C. A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes. *ACM Trans. Softw. Eng. Methodol.* **2012**, *21*, 1–34. [CrossRef]
7. Marcus, A.; Poshyvanyk, D. The Conceptual Cohesion of Classes. In Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), Budapest, Hungary, 26–29 September 2005.
8. Domingos, S.A.M. Study on the Relationships between Cohesion and Coupling Metrics on Fault Prediction in Object Oriented Systems. Master's Thesis, Federal University of Santa Catarina, Florianópolis, Brazil, 2018.
9. Chidamber, S.R.; Kemerer, C.F. Towards a Metrics Suite for Object Oriented Design. *SIGPLAN Not.* **1991**, *26*, 197–211. [CrossRef]
10. Kirgil, E.N.H.; Ayyildiz, T.E. Analysis of Lack of Cohesion in Methods (LCOM): A Case Study. In Proceedings of the 2nd International Informatics and Software Engineering Conference (IISEC), Ankara, Turkey, 16–17 December 2021.
11. Li, W.; Henry, S. Maintenance Metrics for the Object Oriented Paradigm. In Proceedings of the 1st International Software Metrics Symposium, Baltimore, MD, USA, 21–22 May 1993.
12. Saadati, M.; Motameni, H. Measuring Cohesion and Coupling of Object-Oriented Systems. *J. Math. Comput. Sci.* **2014**, *9*, 149–156. [CrossRef]
13. Henderson-Sellers, B.; Constantine, L.L.; Graham, A. Coupling and Cohesion (towards a Valid Metrics Suite for Object oriented Analysis and Design). *Object Oriented Syst.* **1996**, *3*, 143–158.
14. Bieman, J.M.; Kang, B.-K. Cohesion and Reuse in an Object-Oriented System. *ACM SIGSOFT Softw. Eng. Notes* **1995**, *20*, 259–262. [CrossRef]
15. Alzahrani, M.; Melton, A. Defining and Validating a Client-Based Cohesion Metric for Object-Oriented Classes. In Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 4–8 July 2017.
16. Ganesh, S.H.; Raj, H.B.V. Performance Based Analysis on MALCOM—A Software Metric. In Proceedings of the 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], Nagercoil, India, 19–20 March 2015.
17. Wanjiku, R.; Okeyo, G. Scoped Class Cohesion Metric for Software Process Assessment. *Int. J. Comput. Sci. Issues* **2016**, *13*, 12–18.
18. Chen, C.-Y.; Tai, K.-Y.; Chong, S.-S. Quality Evaluation of Structural Design in Software Reverse Engineering: A Focus on Cohesion. *IEEE Access* **2021**, *9*, 109569–109583. [CrossRef]
19. Freeplane. Available online: https://sourceforge.net/projects/freeplane/ (accessed on 10 January 2023).
20. Islam, M.M.; Iqbal, H.; Haque, M.R.; Hasan, M.K. Prediction of Breast Cancer Using Support Vector Machine and K-Nearest Neighbors. In Proceedings of the 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, Bangladesh, 21–23 December 2017.
21. Kotsiantis, S.B. Decision Trees: A Recent Overview. *Artif. Intell. Rev.* **2013**, *39*, 261–283. [CrossRef]
22. Yılmaz, H. Random Forests Yönteminde Kayip Veri Probleminin Incelenmesi ve Sağlik Alaninda bir Uygulama, T.C. Master's Thesis, Eskişehir Osmangazi Üniversitesi, Eskişehir, Turkey, 2014.
23. Ramchoun, H.; Amine, M.; Idrissi, J.; Ghanou, Y.; Ettaouil, M. Multilayer Perceptron: Architecture Optimization and Training. *Int. J. Interact. Multimed. Artif. Intell.* **2016**, *4*, 26. [CrossRef]
24. Brereton, R.G.; Lloyd, G.R. Support Vector Machines for Classification and Regression. *Analyst* **2010**, *135*, 230–267. [CrossRef]
25. Barbur, V.A.; Montgomery, D.C.; Peck, E.A. Introduction to Linear Regression Analysis. *Statistician* **1994**, *43*, 339. [CrossRef]
26. Chatterjee, S.; Hadi, A.S.; Price, B. Regression Analysis by Example. *J. Am. Stat. Assoc.* **2000**, *95*, 1381. [CrossRef]
27. Kulkarni, E.G.; Kulkarni, R.B. WEKA Powerful Tool in Data Mining. *Int. J. Comput.* **2016**, *975*, 8887.
28. Russell, I.; Markov, Z. An Introduction to the Weka Data Mining System. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, WA, USA, 8–11 March 2017; ACM: New York, NY, USA, 2017.
29. Srivastava, S. Weka: A Tool for Data Preprocessing, Classification, Ensemble, Clustering and Association Rule Mining. *Int. J. Comput. Appl.* **2014**, *88*, 26–29. [CrossRef]

30. Botchkarev, A. A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms. *Interdiscip. J. Inf. Knowl. Manag.* **2019**, *14*, 45–76. [CrossRef]
31. Chai, T.; Draxler, R.R. Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)?—Arguments against Avoiding RMSE in the Literature. *Geosci. Model Dev.* **2014**, *7*, 1247–1250. [CrossRef]
32. Feng, C.H.; Conlin, C.C.; Batra, K.; Rodríguez-Soto, A.E.; Karunamuni, R.; Simon, A.; Kuperman, J.; Rakow-Penner, R.; Hahn, M.E.; Dale, A.M.; et al. Voxel-Level Classification of Prostate Cancer on Magnetic Resonance Imaging: Improving Accuracy Using Four-Compartment Restriction Spectrum Imaging. *J. Magn. Reson. Imaging* **2021**, *54*, 975–984. [CrossRef]
33. Li, X.; Mousavi, S.M.; Dadashova, B.; Lord, D.; Wolshon, B. Toward a Crowdsourcing Solution to Identify High-Risk Highway Segments through Mining Driving Jerks. *Accid. Anal. Prev.* **2021**, *155*, 106101. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.