Unsupervised Mapping of Quantitative Measures to Qualitative Characteristics in Hierarchical Software Quality Assurance

Kaveen Liyanage*, Ethan L. Gerard*, Derek Reimanis†, Ann Marie Reinhold †§, Clemente Izurieta†‡§, Brock J. LaMeres* and Bradley M. Whitaker*¶

*Department of Electrical and Computer Engineering, Montana State University, Bozeman, MT, USA
†Software Engineering and Cybersecurity Laboratory, Montana State University, Bozeman, MT, USA
‡Idaho National Laboratory, Idaho Falls, ID, USA
§Pacific Northwest National Laboratory, Richland, WA, USA
¶Corresponding Author: bradley.whitaker1@montana.edu

***Abstract*—Ensuring software quality is essential to the software development and deployment process. Evaluating software for known vulnerabilities and weaknesses is a method of assessing the quality of software. However, the existence of a large number of vulnerabilities and weaknesses can hinder decision-making and mitigation. To address this, hierarchical software quality models aggregate quantitative weakness measures into qualitative characteristics to simplify decision making. However, challenges exist when mapping quantitative measures to qualitative characteristics, especially when the relationship between measures and characteristics is ill-defined/unknown or when prior knowledge is absent, thus posing threats to the construct validity of the mapping. This paper presents a pseudo-label-based regression framework to generate qualitative values when given a set of quantitative measurements. To exemplify this research, we created a use case where we applied the framework to binary program analysis.**

***Index Terms*—Software Quality, Quality Assurance, Machine Learning**

I. Introduction

There has been growing interest in quality-aware software development in recent years due to the threat of cybersecurity vulnerabilities in critical infrastructures. The quality of the software must be an integral part of software development and deployment. Several frameworks have been proposed to address software quality evaluation [1]. The "ISO/IEC 25000 - System and Software Quality Requirements and Evaluation (SQuaRE)" is a series of standards and a theoretical framework for software quality assurance. Likewise, the ISO 25010 standard [2] provides the structure and definitions of the quality characteristics.

Figure 1 shows a simplified view of the abstract levels of a generic Hierarchical Software Quality Model (HSQM). At the base, low-level measures provide a quantitative assessment of a specific area of interest. In the middle, a set of high-level qualitative characteristics is presented. At the top of the model tree, the total quality index provides an overall score. The number of abstract layers may vary depending on the quality model, the domain, and the levels of abstraction chosen by the modeler. Only a few operationalized models exist for practical use in the software development process; Q-rapids [4], VALUE [5], QUAMOCO [6], QATCH [7], and SQALE [8] are some example frameworks. Comparison of these frameworks has also exposed the benefits and weaknesses [9] of these models. The Platform for Investigative software Quality Understanding and Evaluation (PIQUE) [3], [10], [11] is a modern generalized model that mitigates some of the shortcomings of earlier models. One such shortcoming is the mapping of quantitative measures to qualitative characteristics. The relative weighting of the contributions that sub-characteristics have on higher-level characteristics was traditionally performed manually, while PIQUE introduces a data-based approach for this mapping.

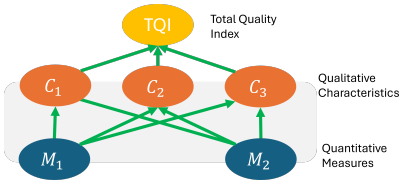

Fig. 1. Simplified HSQM structure. The bottom row shows a collection of quantitative measures. The middle row shows a collection of qualitative characteristics. The top row shows the Total Quality Index (TQI). The arrows show the direction of information flow for aggregation.

Data-based methods are most effective when prior training data is available with qualitative characteristic scores (labels) [4]. However, when developing a new quality model, prior scores (labels) are not available. For example, one objective of software quality may involve improving "Integrity". Labels for training software packages with respect to "Integrity" may be unavailable and/or subjective. Hence, a method is needed

This research was conducted with support from the U.S. Department of Homeland Security (DHS) Science and Technology Directorate (S&T) under contract 70RSAT22CB0000005. Any opinions contained herein are those of the authors and do not necessarily reflect those of DHS S&T.

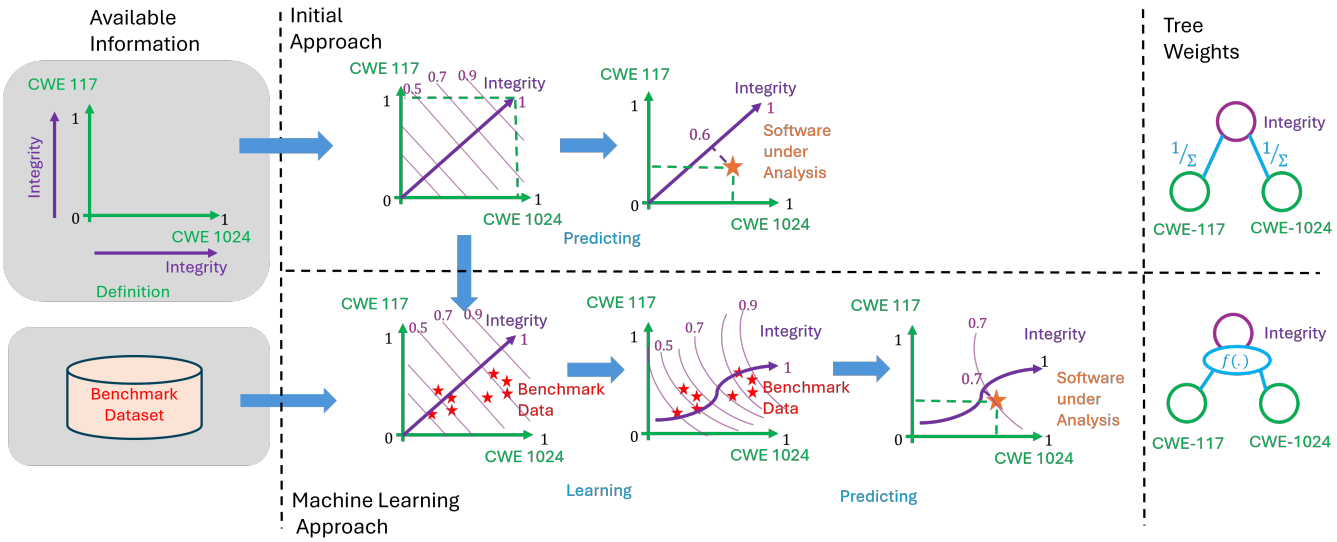

Fig. 2. Workflow of Initial approach and Machine Learning approach from Izurieta et al. [3], for the specific example of mapping CWE 117 and CWE 1024 to the quality aspect "Integrity." Measures CWE-117 and CWE-1024 are drawn in the axes in an abstract space, and the quality aspect (Integrity) is represented as a function in that space. The left column shows the available data, the middle column shows the two approaches, and the right column shows the HSQM tree structure. The Machine Learning approach considers the relative measure scores from a benchmark dataset when determining the qualitative characteristic score of a new software program.

to calculate qualitative characteristic scores in the absence of prior training label data. Regression is a common machine learning (ML) technique that utilizes training data with known labels to predict scores for new data. Using regression in the absence of labels for mapping scores was first introduced as a theoretical framework in the PIQUE model [3]. This paper provides the experimental results of applying the proposed theoretical machine learning framework with examples. Several regression models were tested on a benchmark dataset to evaluate their effectiveness.

II. Background

Our work focuses on contributing an approach to help with the mapping between measures and characteristics. Although definitions of measures usually describe "which" characteristics they affect, definitions fail to explain "how much" a measure affects a characteristic. This problem becomes even worse when multiple measures affect a single characteristic. For example, consider two software weakness measures (i.e., CWE-117 and CWE-1024) used in the PIQUE-bin[1] model. According to the definition from the MITRE, Common Weaknesses Enumeration(CWE) [2], these weaknesses affect the software security characteristic "Integrity." The definition only states that higher counts of CWE-117[3] and CWE-1024[4] should be correlated with the "Integrity" characteristic. However, the definition does not state how much each CWE should affect the "Integrity" characteristic.

[1]https://github.com/MSUSEL/msusel-pique-bin-docker
[2]https://cwe.mitre.org/about/new_to_cwe.html
[3]https://cwe.mitre.org/data/definitions/117.html
[4]https://cwe.mitre.org/data/definitions/1024.html

Several previous methods have been proposed to address this issue in the context of value-based predictive maintenance [12]. The most common approach is an aggregated sum of measures. The weights are calculated either manually or (more commonly) using Bayesian Networks [13] based on the weighted sum algorithm (WSA) [14]. Measure probabilities are calculated using historical data, and the characteristic value is inferred using the Bayesian rule. Reinhold et al. [15] also used probability density functions to calculate more accurate scores that mitigate the effects of variability associated with aggregation. However, most applications calculate probabilities according to a set of discrete value bins (e.g., Low, Medium, and High) and with prior expert knowledge.

III. Approach

Our approach provides alternative methods to generate relationships using a benchmark dataset with minimum prior knowledge transfer, especially when a new quality model needs to be developed. Our approach provides a straightforward setup with a semi-automated process. The complete source code of the experiment is available at https://github.com/MSUSEL/msusel_pique_ML.

The proposed framework consists of two approaches. The initial approach involved pseudo-label generation based on prior knowledge and assumptions. The second is a machine learning approach: training a regressor with a benchmark dataset. Figure 2 provides an overview of the framework from Izurieta et al. [3]. The available information is displayed in the left panel, which includes the definitional relationship of the measures to the characteristics and benchmark dataset from the project domain. The top middle section shows the initial approach with an equal weighting approach. The bottom

middle section shows how the initial approach can be supplemented with a benchmark dataset. Then, pseudo-labels can be generated to train a regressor that will better fit the benchmark data. The learned regressor can be used to predict characteristic values for new projects under analysis. The right panel shows how the two approaches structurally relate to an HSQM. Note that the Machine Learning approach considers the relative measure scores from a benchmark dataset when determining the qualitative characteristic score of a new software program.

A. Problem Formulation

Assume there is a set of n measures: $[M_1, M_2, \ldots, M_n]$. Each measure has a corresponding node value: $[\alpha_1, \alpha_2, \ldots, \alpha_n]$. We assume all reported measure values are between zero and one ($\alpha_i \in [0, 1]$). Finally, we consider a set of m qualitative characteristics: $[C_1, C_2, \ldots, C_m]$. Each characteristic also has a corresponding node value: $[y_1, y_2, \ldots, y_m]$ with $y_j \in [0, 1]$. Now, let $[P^1, P^2, \ldots, P^k]$ be a set of k benchmark projects in the respective domain. When we analyze the set of projects through analysis tools and retrieve quantitative results for n measures and m characteristics, the results can be summarized as shown in Table I.

TABLE I
MEASURE VALUES FOR THE SET OF PROJECTS IN BENCHMARK DATASET

	Measures						Characteristics label (Y)			
	M_1	M_2	M_3		M_n		C_1	C_2		C_m
P^1	α_1^1	α_2^1	α_3^1		α_n^1		y_1^1	y_2^1		y_m^1
P^2	α_1^2	α_2^2	α_3^2		α_n^2		y_1^2	y_2^2		y_m^2
$\vdots$										
P^k	α_1^k	$\ldots$	$\ldots$		α_n^k		y_1^k	y_2^k		y_m^k

In normal regression operations, a set of training labels Y would be provided, which associates each project P^k with characteristics C_j with a value y_j^k. With this information, we can train a regression function for each Characteristic: $\{f_1, f_2, \cdots, f_m\}$ However, in practice, these label values are unavailable due to a lack of previous knowledge. Thus, we first need to estimate a pseudo-label for each project and each Characteristic node value ($\tilde{y}_j^k$). The pseudo-labels, along with the benchmark projects, measure node values, and measure-characteristic definitions are used to generate the regression function for each characteristic (f_j).

B. Pseudo-label assignment

As an initial approach, the weights are assigned manually by analyzing the definition of each measure or product factor and creating a binary indicator function connecting each measure with each characteristic: $I_{i,j}$ is 1 if M_i is related to C_j, otherwise, it is 0. The pseudo-label is then given as the average value of the *relevant* measure node values:

$$\tilde{y}_j^k = \frac{\sum_{i=1}^{n} I_{i,j}\alpha_i^k}{\sum_{i=1}^{n} I_{i,j}}. \quad (1)$$

For the application of software quality, a linear relationship is justified, as improvements in each of the measures lead to an improvement in the characteristic. Although we realize some CWEs may affect quality aspects to varying degrees, this estimate is generally accurate, as even if some CWEs affect quality aspects more and some less, the mean of these values will only vary slightly.

C. Regressor training

In the regression approach, we supplement the initial approach with a representative benchmark dataset in the application domain to establish an informed relationship between measures and characteristics. We look at the data distribution of the benchmark dataset and update the weights from the initial model to fit the distribution. Thus, the regression output scores are relative to the benchmark repository and better represent the project under analysis. This allows us to capture the dependencies between the measures and identify trends in practical implementations. We also normalize the regression function to force the output scores to be between 0 and 1.

In summary, in order to train a regression function f_j for characteristic C_j, we need:

- the node values for all measures and all projects: the α's
- the pseudo-labels for C_j: $[\tilde{y}_j^1, \ldots, \tilde{y}_j^k]$
- the binary indicator between all measures and C_j: $[I_{1,j} \ldots I_{n,j}]$.

After training, the function f_j will take the measure values α^{new} for a new, non-benchmark project P^{new} and return a characteristic value for that project: $\hat{y}_j^{new}$.

IV. EXPERIMENTS AND RESULTS

To evaluate the effectiveness of different regressor models, we experimented with several regressor models. This experiment utilizes a library of Scikit-Learn [16] regression models and *GridSearchCV* for cross-validation. A benchmark repository of training data is used to train the regression models.[5] The training data includes hundreds of example programs and their corresponding CWE values that were yielded from statistical analysis tools. The programs are a collection of binary files from a Kali-Linux operating system.[6] Detailed discussion about the benchmark data, the static analysis tools, and the measure value calculation has been conducted previously [3], [11], [15]. For this study, the measures with constant values were not considered as they do not provide additional information.

Although characteristics are correlated, when evaluating, each quality characteristic can be assumed as independent of each other. Hence, a separate model, f_j, was trained for each characteristic to ensure precision and accuracy in the final quality score. Each measure (CWE value) impacting a characteristic ($I_{i,j} \geq 0$) is fed to the corresponding model, where the model is trained to predict a $\hat{y}_j$ value. The quality characteristics considered were "Access Control", "Confidentiality", "Integrity", "Availability", "Authorization", "Non-repudiation", and "Other". In this application, a value of

[5]https://github.com/MSUSEL/benchmarks
[6]https://www.kali.org/

TABLE II
PROJECT QUALITY CHARACTERISTICS VALUES FROM PSEUDO-LABELS AND RANDOM-FOREST REGRESSOR

Programs	Access Control		Confidentiality		Integrity		Availability		Authorization		Non-repudiation		Other	
	Pseudo	Reg.	Pseudo	Reg.	Pseudo	Reg.	Pseudo	Reg.	Pseudo	Reg.	Pseudo	Reg.	Pseudo	Reg.
tr	0.5418	0.543	0.5631	0.559	0.5625	0.552	0.5413	0.391	0.5472	1.0	0.5560	0.639	0.5587	0.596
pmspike	0.4739	0.283	0.5111	0.362	0.5135	0.357	0.4726	0.166	0.5472	1.0	0.5560	0.639	0.4780	0.29
mpstat	0.5086	0.416	0.5570	0.540	0.5567	0.533	0.5355	0.369	0.5472	1.0	0.5560	0.639	0.5426	0.255
xgc	0.5418	0.543	0.5822	0.636	0.5809	0.628	0.5596	0.452	0.5472	1.0	0.5560	0.639	0.5651	0.620

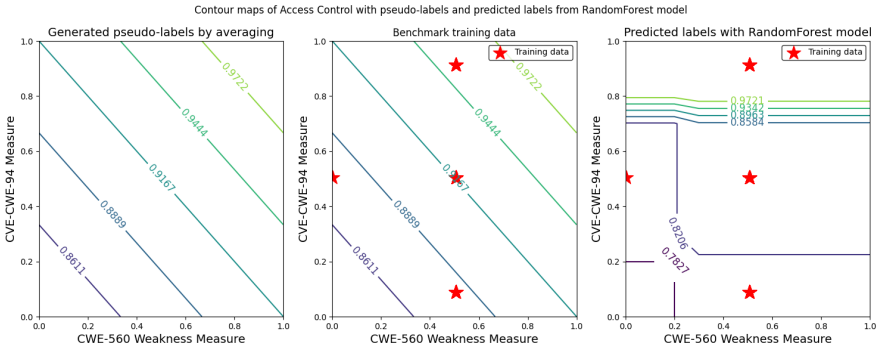

Fig. 3. The left figure shows the pseudo-label contour lines generated for access control by averaging the input features for the quality characteristic "Access Control" in the input space of CWE-94 and CWE-560 measures. The middle figure shows the scattered benchmark projects used for training a regressor with the pseudo labels. In the right figure, contour lines show the learned Random Forest regressor label values. The contour lines shown here are generated assuming the rest of the measure values are one.

0 is considered "bad", and a value of 1 is considered "good" for both measures and characteristics.

Before training the model, a grid search was conducted on the training dataset, yielding suitable hyperparameters. After training the model, it was tested with twenty percent of the dataset, and an R^2 value was calculated.

An example output of initial pseudo-labels and the Random Forest regressor predictions for the corresponding quality characteristics are shown in Table II. Recall that the pseudo-labels ($\tilde{y}_j^k$) are generated using the linear model in Eq. 1. In contrast, the regression-generated labels ($\hat{y}_j^k$) use the linear model as a baseline but also incorporate information about the benchmark dataset into the prediction.

V. DISCUSSION

Experimental results show the validity of the proposed theoretical model of using pseudo-labels for mapping quantitative measures to qualitative characteristics. Some regression models discussed in this paper are effective at incorporating information from a benchmark dataset to consider quality characteristic values. Rather than simply averaging the results from the measure values, the regression models have the ability to perform a meaningful comparison between a software under analysis and a corpus of benchmark data. Our results show the viability of the proposed theoretical framework on real-world data with different regressor models when training labels are unavailable.

First, as an example, consider the quality characteristic "Access Control" for the PIQUE-Bin model. Twelve measures (CWE-560, CWE-94, CWE-191, CWE-290, CWE-190, CWE-332, CWE-59, CWE-273, CWE-426, CWE-190, CWE-310 and CWE266) are identified as related to "Access control" by evaluating the CWE definitions. The base assumption is that "CWE measure values are linearly related to the Access Control score".

Figure 3 (left panel) shows how the contour lines were generated by the initial approach. Note that the contour lines are calculated for the given input space by considering all the other input feature values to be set to one. In other words, if all the other features are equal to one, how does changing CWE-94 and CWE-560 affect the "Access Control" score? The initial method of averaging creates parallel contour lines with a slope of -1. At the top-right point of the plot (where CWE-94 and CWE-560, in addition to the other 10 CWEs that are not visualized, have a score of 1), the "Access Control" score is also 1. As either CWE-94 or CWE-560 decreases, the "Access Control" score also decreases, in a linear fashion.

Figure 3 (middle panel) shows the CWE-94 and CWE-560 measure values for four benchmark training projects. Since the "Access Control" labels for these benchmark projects do not exist, we must use an initial approach to generate pseudo-label values.

Figure 3 (right panel) shows the training dataset and the learned Random Forest regressor contour lines for the quality characteristic "Access Control". The regressor has contour lines that better reflect the CWE distribution of the benchmark data. It can be seen that the sensitivity of "Access Control" to CWE-560 is minimum or almost constant. This is because the majority of the benchmark data have the same CWE-560 value. Because of this, if a new software under analysis has a different CWE-560 score, the regression algorithm has greater

TABLE III
REGRESSION MODELS R-SQUARED VALUES

Model	Access Control	Confidentiality	Integrity	Availability	Authorization	Non-rep.	Other
Linear	1.000	0.904	0.640	0.3573	1.000	1.000	1.000
Ridge [17]	1.000	0.999	0.999	0.999	1.000	1.000	1.000
SGD [18]	0.976	0.994	0.994	0.994	0.397	0.9145	0.990
Decision-Tree [19]	0.943	0.862	0.819	0.916	1.000	0.961	0.981
Lasso [17], [20]	1.000	0.998	0.999	0.997	1.000	1.000	1.000
Elastic-Net [17]	1.000	0.998	0.998	0.999	0.993	0.999	1.000
Random-Forest [21]	0.943	0.890	0.872	0.949	0.982	0.956	0.982
Extra-Trees [19], [22]	0.951	0.905	0.889	0.963	1.000	0.961	0.959
Kernel-Ridge [23]	1.000	1.000	1.000	1.000	1.000	1.000	1.000
OMP [24]	0.737	0.563	0.618	0.641	1.000	0.913	0.632
K-NN [25]	0.943	0.920	0.903	0.964	1.000	0.967	0.967
Gradient Boost [26]	0.958	0.933	0.919	0.969	-0.024	0.9696	0.986

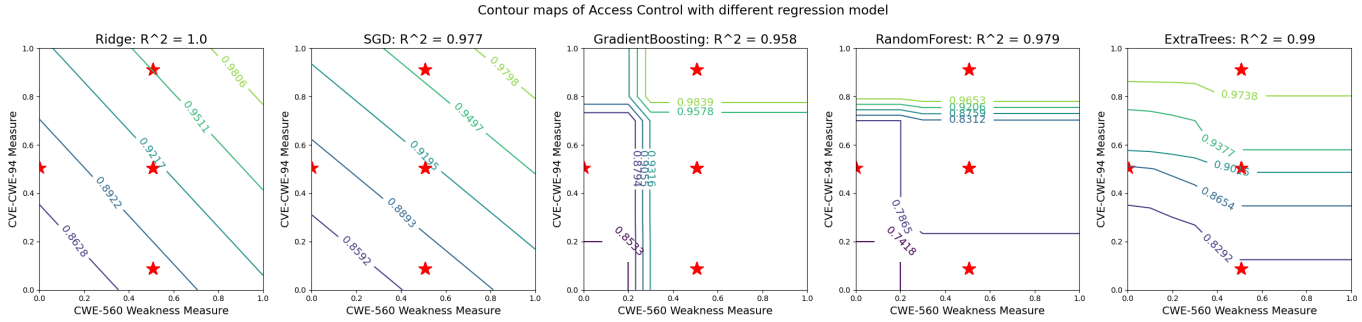

Fig. 4. The left figure shows contour lines generated by different regressor models in the input features for the quality characteristic "Access Control" in the input space of CVE-CWE-94 and CWE-560 measures. The Ridge, SGD, Gradient Boost, Random Forest, and Extra Trees regressors with their R^2 scores are shown.

uncertainty with respect to its impact on "Access Control." The CWE-94 measure has a slightly higher sensitivity on the characteristic score because the benchmark dataset has greater variance for this measure. Thus, the magnitudes of the contour lines are changing with the CWE-94 changes. This shows that the regression model fine-tunes the output predictions to match the distribution of the benchmark dataset.

The results for each model's R^2 score are shown in Table III. It was observed that when a regression model has an R^2 value of 1, the model ignores the data and mostly imitates the initial pseudo-label assumption. Hence, a model with a slightly lower R^2 value strikes a balance between holding true to the initial assumption and adapting to the benchmark data. Figure 4 shows the contour lines of different regressors and their respective R^2 values. It should be noted that Ridge and SGD are linear models, and the others are nonlinear models. The linear models with $R^2 = 1$ always fit to the pseudo-labels; $R^2 < 1$ indicates that the models learn a different weighting. For the nonlinear models, the R^2 value is almost always less than 1. Note that all the nonlinear models have identified that a change of CWE-560 after a value of 0.5 does not contribute to the characteristic score. This is justified as we do not have any benchmark data to support the assumption that a higher CWE-560 measure value could have an impact on the characteristic score. Hence, the predicted characteristic score will provide insight into the quality of the project under analysis with respect to the known benchmark project corpus.

In Table III, the "Authorization" characteristic has the most inconsistent behavior across regression models. When investigated, only a single measure was associated with authorization. In addition, that particular measure had little variance in the benchmark dataset, making it difficult for a regression algorithm to generalize in a nonlinear fashion. This indicates the distribution of the projects for each measure has an impact on the performance of the regressor. In summary, the results show that the proposed framework was able to achieve a balance between the initial assumptions and the benchmark data. The selection of the regressor is dependent on the application and the intended simplicity of the overall quality score. However, a selection criterion has yet to be developed to identify the best-suited regression model for the application.

By observing the results, several future directions were identified for the practical application of the proposed framework. First, a metric needs to be introduced to evaluate the regression models' prediction "balance" between the pseudo-labels and benchmark data. This will help modelers to decide which regression models to use and how much a regressor is "balanced". Second, a metric to asses the quality of the input benchmark dataset must be developed. This will give users confidence in the predicted score with respect to the benchmark dataset. Finally, a guideline will be developed on how to interpret the generated results and use the scores for decision-making. This will help in identifying trends and common practices in the application domain.

VI. Threats to Validity

Several identified threats to validity must be considered when using regression methods for node value predictions. An internal threat to validity occurs when regression models do not guarantee a monotonically increasing function. To mitigate this threat, we are exploring a post-processing function to force the regression output to monotonically increase. Also, if a new program has vastly different measure values than the benchmark data distribution, the model will try to provide a score that closely matches a benchmark datum. This could obfuscate the new project's difference in the characteristic score prediction. To mitigate this, highlighting the measure input value differences will allow users to conduct further analysis.

External threats to validity exist because the developed method was only tested with the PIQUE-bin benchmark data, and the results cannot be generalized. We are planning to test with other PIQUE models. The effects of the benchmark dataset are still unknown, so metamorphic testing on the ML model is planned. In particular, the impact of identical CWE findings and constant CWE values must be studied further. Currently, CWEs that produce constant findings are omitted from the regression model.

As a threat to construct validity, the initial assumption of a linear relationship and the selection of the regression model may lead to variations in the model's behavior. To mitigate this threat, selection criteria will be developed for the users.

VII. Conclusion

This paper presents a framework that maps low-level quantitative measures to high-level qualitative characteristics. A pseudo-label-based regression method is presented. First, the initial weighting of the manual assignment is used to estimate characteristic values (pseudo-labels). These pseudo-labels, in combination with a benchmark dataset, are used to train the regression model. The trained regression model was used to predict updated characteristic values. Each characteristic was predicted using an independently trained regressor. Several linear and nonlinear regression models were evaluated. The framework is discussed, and several threats to validity were identified. The main benefits of this proposed framework are (1) the input and output can easily be compared using the regression model and (2) the regression model considers both the natural linearity assumption and the quality values of a benchmark dataset when making characteristic value predictions. In the future, this framework will be integrated into PIQUE as the default method with an option for the user to customize parameters.

References

[1] S. Martinez-Fernandez, A. M. Vollmer, A. Jedlitschka, X. Franch, L. Lopez, P. Ram, P. Rodriguez, S. Aaramaa, A. Bagnato, M. Choras, and J. Partanen, "Continuously assessing and improving software quality with software analytics tools: A case study," *IEEE Access*, vol. 7, pp. 68219–68239, 2019.

[2] ISO, "ISO/IEC 25010 systems and software engineering-systems and software quality requirements and evaluation (square)-product quality model," 2023.

[3] C. Izurieta, D. Reimanis, E. O'Donoghue, K. Liyanage, A. R. Manzi Muneza, B. Whitaker, and A. M. Reinhold, "A generalized approach to the operationalization of software quality models," *PeerJ Computer Science*, vol. 10, p. e2357, Oct. 2024.

[4] X. Franch, L. Lopez, S. Martínez-Fernández, M. Oriol, P. Rodríguez, and A. Trendowicz, *Quality-Aware Rapid Software Development Project: The Q-Rapids Project*, pp. 378–392. Springer International Publishing, 2019.

[5] E. Mendes, P. Rodriguez, V. Freitas, S. Baker, and M. A. Atoui, "Towards improving decision making and estimating the value of decisions in value-based software engineering: the value framework," *Software Quality Journal*, vol. 26, pp. 607–656, Mar. 2017.

[6] S. Wagner, A. Goeb, L. Heinemann, M. Kläs, C. Lampasona, K. Lochmann, A. Mayr, R. Plösch, A. Seidl, J. Streit, *et al.*, "Operationalised product quality models and assessment: The quamoco approach," *Information and Software Technology*, vol. 62, pp. 101–123, 2015.

[7] N. M. Villegas, H. A. Müller, G. Tamura, L. Duchien, and R. Casallas, "A framework for evaluating quality-driven self-adaptive software systems," in *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ICSE11, pp. 80–89, ACM, May 2011.

[8] J.-L. Letouzey and T. Coq, "The sqale models for assessing the quality of real time source code," *Online,[Accessed: 17.07. 2017]*, pp. 1500303973–1157276278, 2010.

[9] C. Izurieta, I. Griffith, and C. Huvaere, "An industry perspective to comparing the sqale and quamoco software quality models," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 287–296, IEEE, Nov. 2017.

[10] D. M. Rice, *An extensible, hierarchical architecture for analysis of software quality assurance*. PhD thesis, Montana State University-Bozeman, College of Engineering, 2021.

[11] A. L. Johnson, "The analysis of binary file security using a hierarchical quality model," Master's thesis, Montana State University-Bozeman, College of Engineering, 2022.

[12] D. Sanchez-Londono, I. Roda, and G. Barbieri, "The requirements of a value model for the strategic implementation of predictive maintenance," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1276–1281, 2023.

[13] M. Manzano, E. Mendes, C. Gómez, C. Ayala, and X. Franch, "Using bayesian networks to estimate strategic indicators in the context of rapid software development," in *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, PROMISE'18, pp. 52–55, ACM, Oct. 2018.

[14] B. Das, "Generating conditional probabilities for bayesian networks: Easing the knowledge acquisition problem," 2004.

[15] A. M. Reinhold, B. Boles, A. R. M. Muneza, T. McElroy, and C. Izurieta, "Surmounting challenges in aggregating results from static analysis tools," *Military Cyber Affairs*, vol. 7, no. 1, pp. 5–11, 2024.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, 2010.

[18] W. Xu, "Towards optimal one pass large scale learning with averaged stochastic gradient descent," 2011.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*. Routledge, Oct. 2017.

[20] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale -regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, pp. 606–617, Dec. 2007.

[21] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[22] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, Mar. 2006.

[23] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: The MIT Press, 2012.

[24] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[25] S. M. Omohundro, "Five balltree construction algorithms," 2009.

[26] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, Oct. 2001.