Comparing Multivariate Time Series Analysis and Machine Learning Performance for Technical Debt Prediction:

The SQALE Index Case

Mikel Robredo[1], Nyyti Saarimäki[2], Rafael Peñaloza[3], Davide Taibi[1], Valentina Lenarduzzi[1]
[1] University of Oulu — [2] Tampere University — [3] University of Milano-Bicocca
mikel.robredomanero@oulu.fi; nyyti.saarimaki@tuni.fi; rafael.penaloza@unimib.it
davide.taibi@oulu.fi; valentina.lenarduzzi@oulu.fi

ABSTRACT

Predicting Technical Debt has become a popular research niche in recent software engineering literature. However, there is no consistent approach yet that succeeds in entirely capturing the nature of this type of data. We applied each technique on a dataset consisting of the commit data of a total of 28 Java projects. We predicted the future values of the SQALE index to evaluate their predictive performance. Using these techniques we confirmed the predictive power of each of them with the same commit data. We aim to investigate further the time-dependent nature of other types of commit data to validate the existing prediction techniques.

KEYWORDS

Technical Debt, Software Quality Mining Software Repositories, Empirical Software Engineering

ACM Reference Format:
Mikel Robredo[1], Nyyti Saarimäki[2], Rafael Peñaloza[3], Davide Taibi[1], Valentina Lenarduzzi[1]. 2024. Comparing Multivariate Time Series Analysis and Machine Learning Performance for Technical Debt Prediction:: The SQALE Index Case. In *International Conference on Technical Debt (TechDebt '24), April 14–15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3644384.3644472

1 INTRODUCTION

In Software Engineering, and especially in Mining Software Repository studies researchers often investigate the relationships among different variables collected from the history of software projects. As an example, researchers have investigated correlations between two variables such as code smells [6, 8], their trend over time [4], and the impact of different software qualities [2, 5, 9]. However, a large number of these studies have overlooked the limitations of the temporal dependency of the variables, and the possible threats related to the usage of statistical techniques not designed for analyzing temporally dependent data [7]. As an example, the introduction of a technical issue in a commit heavily depends on the code that was present in the repository before the commit. Still, several studies have not considered this aspect, mainly due to a lack of clear statistical techniques that can be used in this context.

TechDebt '24, April 14–15, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0590-8/24/04.
https://doi.org/10.1145/3644384.3644472

Saarimäki et al. [7] highlighted three main issues in previous studies: 1) discarding the temporal nature of the commits, 2) assuming independence in the data, and 3) mixing projects of different sizes, where big projects overwhelm small ones.

To overcome the issues related to temporal dependence, they proposed to analyze dependent data in software engineering considering the variable dependency and the time effect [7].

Technical Debt (TD) stands as an important metric in software projects as it measures the effort that professionals need to put into making the code clean. That is, a higher amount of mistakes in the code concludes in a higher TD to be confronted. In this sense, TD denotes time dependence towards the mistakes made in the past, consequently, data-analysis techniques that assume the time dependence should be considered when analyzing TD.

In this direction, the research community applied traditionally used machine learning models [10, 11]. Similarly, we identified two relevant works that applied time-series analysis to forecast technical debt [3, 12] comparing univariate and multivariate time-series analysis models. The obtained results are promising in terms of prediction performance. We believe that time series analysis techniques form a robust approach to predict time-dependent variables such as TD, and further believe that there exist potential variables that can predict the behavior of TD.

Therefore, we designed and conducted an empirical study to first, compare the prediction performance of the considered time-dependent models, and second compare this model against time-agnostic models already used in the literature such as Machine Learning (ML) models. Specifically, we adopted two different multivariate applications of the Autoregressive Integrated Moving Average (ARIMA) model, and seven ML models (four linear and three non-linear). Unexpectedly, ML models outperform the time-dependent models. The achieved results demonstrated it even if structuring the data into time series provided better results for all the models in this study.

2 APPROACH

Data collection. We considered 28 projects from the Technical Debt Dataset [1] version 2.0. The projects were developed in Java, were older than three years, had more than 500 commits and 100 classes, and usage of an issue-tracking system with at least 100 issues reported. We considered this dataset since all the projects already contained the data related to the SQALE Index.

Preprocessing. Since the data contained 205 potential independent variables to explain the behavior of the SQALE Index, first we performed a *feature selection* process combining commonly used techniques such as *feature importance* and *variance thresholding* among others. Secondly, we prepared the time series data to fit our time-dependent models by *serializing* the original data. For that, we found bi-weekly and monthly series to be the most suitable periodicity levels. There were periods with non-existent original commit data, therefore we overcame it through *linear interpolation* techniques considering neighboring observations. We considered the original data to be used within the ML models as well to study the effect of non-serialized data in the prediction performance.

Data Analysis. On one hand, we implemented two multivariate time-dependent (or time-series) models with the monthly and by-weekly datasets. First, the ARIMAX model stands for the classic ARIMA model with extra independent variables included to predict the behavior of the SQALE index. Second, the ARIMAX + LM model which has been previously adopted in [12], relies on the implementation of a univariate ARIMA model for the prediction of each independent variable's new observations and the subsequent use of these values for the prediction of the SQALE index by performing Linear Regression (LM). On the other hand, we implemented the two serialized datasets and the original dataset with a total of 7 well-known ML models. For a complete comparison we chose 4 linear models (*Multiple Linear Regression* (MLR), *Stochastic Gradient Descent* (SGD), *Lasso Regression* (L1) and *Ridge Regression* (L2)) and 3 non-linear (*Support Vector Machine* (SVM), *Extreme Gradient Boost* (XGB) and *Random Forest Regression* (RF)) models.

Performance Evaluation. For each model, we performed the *Walk-Forward Train-Test* technique which trains and tests a model greedily and respecting the temporal order of the data. We initiated this process by setting 80% of the existing data as training data, then iteratively predicted future values, measured the performance, and added the predicted value into the training set for further predictions. Similarly, we used *Mean Absolute Percentage Error* (MAPE), *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), and *Root Mean Squared Error* (RMSE) as prediction performance metrics.

3 RESULTS

We successfully implemented the proposed prediction models, and further obtained reasonable predictive performance with all of them. Thus, we initially confirmed that time-dependent models can perform similarly to the widely used ML models.

ARIMA+LM overcomes ARIMAX. Results consistently demonstrated a better predictive performance from the ARIMA+LM model than ARIMAX (12% to 21% MAPE with monthly data for instance) in all the performance metrics and with the different serialized datasets. Further, the shorter periodicity level (biweekly) improved the ARIMA+LM model while a longer periodicity (monthly) worked better for the ARIMAX model.

XGB superiority. ML models consistently overcame the predictive performance shown by the time-dependent models. Overall the non-linear models demonstrated to perform much better with XGB providing the best results (1.87%, 1.14% and 1.71% MAPE with monthly, bi-weekly, and original data respectively). Moreover, ML models presented better results with periodically serialized data than with the original data, thus demonstrating the temporal relationship existing in commit data.

Scarce periodicity. Although performance results were improved with the serialized datasets, initial descriptive analysis denoted a lack of periodicity in the original commit data, which resulted in multiple missing observations that had to be interpolated. This scenario suggests that periodically committing code can help create better quality time series data, and therefore improve the predictive performance of both time series and ML models.

4 CONCLUSIONS

We performed an empirical study comparing two multivariate time-series analysis approaches and seven ML models to predict the value of the SQALE index. Three data sets were used to train the models, based on different assumptions about the temporal periodicity (original commits, bi-weekly, and monthly sampling). Their predictive performance was measured over several metrics analyzing 28 open-source Java projects. The results clearly show that ML models outperform the time-series models. However, structuring the data into time series provided better results for all the models in this study. Therefore, although the time-agnostic ML approaches were superior compared to time-dependent approaches, time could have an impact on TD prediction. Out of all of the tested models, the non-linear XGB showed the best performance, while from the time-dependent models combining ARIMA with linear regression provided better results than using ARIMAX.

REFERENCES

[1] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. 2019. The Technical Debt Dataset. In *PROMISE '19*.
[2] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. 2020. Some sonarqube issues have a significant but small effect on faults and changes. a large-scale empirical study. *Journal of Systems and Software* (2020), 110750.
[3] Maria Mathioudaki, Dimitrios Tsoukalas, Miltiadis Siavvas, and Dionysios Kehagias. 2022. Comparing Univariate and Multivariate Time Series Models For Technical Debt Forecasting. In *Computational Science and Its Applications*. 62–78.
[4] S. Olbrich, D. S. Cruzes, V. Basili, and N. Zazworka. 2009. The evolution and impact of code smells: A case study of two open source systems. In *International Symposium on Empirical Software Engineering and Measurement*. 390–400.
[5] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto, and Andrea De Lucia. 2018. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering* 23, 3 (01 Jun 2018), 1188–1221.
[6] Fabio Palomba, Damian Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman, and Alexander Serebrenik. 2018. Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells? *IEEE Transactions on Software Engineering* (2018), 1–1.
[7] Nyyti Saarimäki, Sergio Moreschini, Francesco Lomio, Rafael Penaloza, and Valentina Lenarduzzi. 2022. Towards a Robust Approach to Analyze Time-Dependent Data in Software Engineering. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 36–40.
[8] Tushar Sharma, Paramvir Singh, and Diomidis Spinellis. 2020. An Empirical Investigation on the Relationship between Design and Architecture Smells. *Empirical Software Engineering* 25 (2020).
[9] Dag I. K. Sjoberg, Aiko Yamashita, Bente Anda, Audris Mockus, and Tore Dyba. 2013. Quantifying the Effect of Code Smells on Maintenance Effort. *IEEE Trans. Softw. Eng.* 39, 8 (aug 2013), 1144–1156.
[10] Dimitrios Tsoukalas, Dionysios Kehagias, Miltiadis Siavvas, and Alexander Chatzigeorgiou. 2020. Technical debt forecasting: An empirical study on open-source repositories. *Journal of Systems and Software* 170 (2020), 110777.
[11] Dimitrios Tsoukalas, Nikolaos Mittas, Alexander Chatzigeorgiou, Dionysios Kehagias, Apostolos Ampatzoglou, Theodoros Amanatidis, and Lefteris Angelis. 2021. Machine learning for technical debt identification. *IEEE Transactions on Software Engineering* 48, 12 (2021), 4892–4906.
[12] Ioannis Zozas, Stamatia Bibi, and Apostolos Ampatzoglou. 2023. Forecasting the Principal of Code Technical Debt in JavaScript Applications. *IEEE Transactions on Software Engineering* 49, 4 (2023), 2498–2512.