A Survey on Data Science Techniques for Predicting Software Defects

Farah Atif, Manuel Rodriguez, Luiz J.P. Araújo, Utih Amartiwi, Barakat. J. Akinsanya, and Manuel Mazzara

Abstract In recent years, data science has been used extensively to solve several problems and its application has been extended to several domains. This paper summarises the literature on the synergistic use of Software Engineering and Data Science techniques (e.g. descriptive statistics, inferential statistics, machine learning, and deep learning models) for predicting defects in software. It shows that there is a variation in the use of data science techniques and limited reasoning behind the choice of certain machine learning models but also, in the evaluation of the obtained results. The contribution of this paper has to be intended as a categorization of the literature according to the most used data science concepts and techniques from the perspectives of descriptive and inferential statistics, machine learning, and deep learning. Furthermore, challenges in software defect prediction and comments on future research are discussed and forwarded.

1 Introduction

Software products and platforms have become, today, omnipresent in every domain and an important asset for every company. Despite the evolution of programming languages, coding environments, modern design patterns, and management approaches that aim to facilitate building software with minimum ambiguity and higher reliability. The presence of defects is imminent and can be discovered during the coding or testing phase or later after deployment.

In the literature, the term defect can be found under different naming such us error, fault, and failure. In the work of Mantyla and Lassenius [42] they grouped defect definitions into three categories according to the works in this field. The de-

Farah Atif, Manuel Rodriguez, Luiz J.P. Araújo, Barakat Akinsanya, Manuel Mazzara
Innopolis University, Republic of Tatarstan, Russia,e-mail: f.atif@innopolis.university, e-mail: m.rodriguez.osuna@innopolis.university, e-mail: l.araujo@innopolis.university, e-mail: b.akinsanya@innopolis.university, e-mail: m.mazzara@innopolis.ru

fects can be perceived as failures when they cause the crashing of the system or yield faulty results [42] [32] [54]. From the IEEE Standard failure is defined as ”the inability of a function to meet the expected requirements”. Other defined defects as faults, which are errors in the code but not necessarily lead to system failure [8] [1] [48]. IEEE defines faults as ”incorrect decision taken while understanding the given information, to solve problems or in implementation of process”. Also, defects can be considered as a deviation from quality [17] [33] [25] [49]. From the PSP perspective, the defect is counted when a change in the program occurs.

Data Science (DS) is a scientific field and paradigm that is driving a re-search evolution in statistics, computer science, and Artificial Intelligence (AI). It emphasizes the use of general methods without changing its application, irrespective of the domain. DS aims to meet the challenges of the increasing demand and sustainable future while ensuring the best solutions. Over the years, DS has proved its ability to improve the software engineering process by following methodologies to retrieve useful insights from data [26].

Software Defects Prediction (SDP) in early stages is essential for software reliability, and quality [4]. Detecting defects after the release of the software is an exhausting, and time-consuming process. Hence, the ability to make earlier predictions will help to reduce the effort, and cost of resolving these defects. This survey aims to investigate different Data Science approaches and applications for predicting defects in software.

In this paper, we continue the work initiated in [3] about the investigation of synergistic approaches between Software Engineering and Data Science. Here we do not intend to provide a full Systematic Literature Review, but instead to summarise an overview of the most commonly used techniques. The collection of references and papers have been conducted querying the major research database such as Google Scholar, DBLP, ACM Digital Library, and IEEE explore using generic keywords such as *software defects, descriptive statistics, statistical inference, defects prediction in software, regression models, software quality*. We left a systematic and more comprehensive approach for a future extension of this work.

The remaining of this paper is summarised as follows. Section 2 describes defects, their impact and their nature. Section 3 presents methods employed for predicting defects . Subsections 1 and 2 show the descriptive and statistic inference methods. Section 4 Regression models, Section 5 presents Classification models for predicting defects. A discussion of the main findings from studies is presented in Section 6. Suggestions for future work is shown in Section 7.

2 Predicting Defects

Software defects can be attributed to many factors, the most common one is human error [24], these types of errors can be classified according to their nature, the severity of the impact they can cause and the priority they have in being fixed.

The nature of the defects can be grouped into main categories that include performance errors which are bound to the speed, resource consumption response time and stability of the software [37]. Compatibility defects affect the performance of the software on particular types of devices, operating systems, and hardware [46]. Security defects are related to weaknesses that allow a potential security attack [43]. The severity is measured with a number from one to four. A severity error number one is the highest and corresponds to an outage in the system and considerable impact to the user. The severity error number two also implies damage. However, a temporary solution or a circumvention to this issue is available. The remaining severity errors three and four are representative of lesser damage that can range from a touch and feel problem to an annoyance [52].

As aforementioned, predicting defects at early stages contribute to guarantee the reliability of the software and reduce the costs of maintenance. The process of predicting defects aims to classify software modules into fault-prone and none fault-prone. In the following, we demonstrate data science techniques used for predicting defects in software.

3 Descriptive and Inferential Statistical Methods in Defect Prediction

Statistical methods are important for understanding the data and make the right use of it. In software defect prediction different techniques derived from statistical concepts have been used such as descriptive statistics, and inferential statistics. In this section, we provide the literature review of the techniques used in software DP.

3.1 Descriptive Statistics

Descriptive statistics is a summary statistic used to organize, present, and analyze data. It relies on numerical and graphical techniques [26]. In software defect prediction, descriptive statistic has been used widely to describe the metrics obtained from the software [40] [6] [57] [21] [51] by calculating the mean, the variance, the median, the maximum, and the minimum. The most used metrics in this area are Chidamber and Kemerer (CK) set of object-oriented metrics, lines of code (LOC) metric, McCabe Cyclomatic complexity metric, and Halstead's metrics. Descriptive statistics is also used to understand the dataset in this work [10] the mean of lines of code and the mean of vulnerabilities were calculated for each application in the dataset.

In the works of [11] [7], the mean and the standard deviation have been used for measuring the uncertainty of the predictions. Which is caused by the flipping between cross-validation runs by the classifiers. Metrics used for model evaluation are namely the accuracy, precision, recall, and F-score. This work [7] suggests that the

oscillations in predictions can be caused by the fold generation. Also, non-stratified folding may cause class imbalance, which can significantly affect the performance of the model.

3.2 Statistical Inference Methods

Statistical Inference (SI) is the process of inferring something about a population based on what is measured in the sample. It is divided into two types, parametric and non-parametric. The parametric statistics requires some assumptions about the population parameters. The most used assumption about data is the normal distribution, some papers also considered the homogeneity of variance. In the work of [38], the Shapiro-Wilk test for normality has been used, and Levene's test for homogeneity. If the data does not meet the assumption, then non-parametric statistics should be applied.

Some researchers do such a preliminary process to identify important features and metrics. In this work [58], correlation analysis has been used before building the model to check whether there is a relationship between the number of defects and the metrics. [2] selected k-best features using Chi-square for three datasets. To handle the problem of the non-normal distribution of data, density kernels such as Gaussian, Epanechnikov, Biweight, and Triweight are used in [29]. Furthermore, correlation analysis both parametric (Pearson) and non-parametric (Spearman) supports the evaluation of relationships between metrics, the existence of col-linearity, and evaluate the prediction [39]. For instance, Pearson and Spearman correlations are used to measure the goodness of fit of regression model in [28], [23], [6], and [21]. In [53] correlation between independent values has been used to check collinearity. However, in real data, the correlation between metrics exists and can affect the prediction quality. To cover the multi-collinearity, dimensionality reduction technique called principal component analysis (PCA) was applied for this problem in [58].

Comparative analysis in defect prediction is used to compare models [20] [36], defect metrics, process metrics, and evaluation metrics[38]. The parametric statistics used to compare between two groups are z-test[20] and T-test [36] while Analysis of Variance (ANOVA) used to compare more than 2 groups. In [38] Wilcoxon matched pair, a non-parametric comparative analysis is also used to identify which process metrics improve defect prediction models based on product metrics. In [36], Analysis of Variance (ANOVA) has been used with Post-hoc analysis and Bayesian approach to compare the result of defect prediction between models. To evaluate the model, Area Under ROC Curve (AUC) and an alternative named H measures have been used for model evaluation. These evaluations are compared by using non-parametric statistics called the Friedman test and got no significant difference between them. Moreover, the review analysis in [31] also suggests using effect size for better evaluation.

4 Regression Models

Regression techniques are powerful statistical methods that have been exploited for predictions and in classification. In this section, we present the use of linear regression, logistic regression, and Poisson regression in DP.

4.1 Linear Regression

(LR)Regression methods allow predicting unknown variable based on one or more known variables [21]. It has been used in DP for different purposes and in different ways. In [58] LR was used to predict defects on a dependency graph to identify dependencies between pieces of code, it was also used by [23] to predict defects from defect history in SAP Java code.

4.2 Logistic Regression

Logistic regression is a generalisation of the linear regression to binary classification [45]. In the works of [6] [21] [51] it was used to study the relationship between the metrics and fault-proneness of classes.

It also has been used as a classifier for LSTMs [9] [56] and deep belief network (DBN) [58]. Many works employed logistic regression, which predicts the probability of being a defect. If the predicted probability is greater than a threshold -usually 0.5- they label it as a defect, if it is less than the threshold, then it is labeled as non-defect [30] [45].

The work of [51] held an empirical validation of object-oriented metrics for predicting the fault proneness in the classes on a large dataset. It compared the performance of logistic regression with two machine-learning methods (decision trees (DT) and artificial neural networks (ANN)).

With a different approach in [9] Random Forests (RF) outperformed significantly logistic regression on Samsung dataset. However, logistic regression performed better than RF on smaller dataset Promise.

In the work of [21] both linear and logistic regressions were used, univariate and multivariate on CK metrics. Univariate regression analysis is used to examine the effect of each metric separately, while multivariate regression analysis examines the common effects of the metrics in fault-proneness detection [21]. They used both regressions because logistic regression only classifies a class into a defect or not. However, it is necessary to predict the number of bugs in a class, that is why linear regression has been used to predict the number of bugs.

4.3 Poisson Regression

Poisson regression is another type of regression that is used to model counted data and contingency tables. In DP, it can be used for small components of software [12]. It was observed that Poisson provides an estimate of several defects in a small component unlike techniques such as discriminant analysis which identifies components that are likely to have defects with a limited number of defects. The methodology consists of calculating the probability of defects regarding an expected number of defects using a defined set of metrics.

5 Classifying Software Defects Using Machine Learning & Deep Learning

Classification models aim to classify software modules into defect-prone and non-defect-prone either using software metrics or analyzing log files. In the following section, we demonstrate the use of machine learning and deep learning models for software DP.

5.1 Machine Learning Models

The result of recent research using machine learning and deep learning models for DP tasks has shown promising results. The most commonly used ML models for DP tasks are Random Forest (RF), Support Vector Machine (SVM), Naive Bayes (NB), Logistic Regression, K Nearest Neighbourhood (KNN), Bayesian Networks (BN), Radial basis function (RBF), Decision Tree (DT) and the most commonly used DL models for DP tasks are Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Long short-term memory (LSTM), Multi-Layer Perceptron (MLP).

The work of [14] used weighted SVM to handle the problem of class imbalance.

Several researchers have used SVM with different software metrics. In the work of [41], SVM was used on CK metrics and on McCabe Halstead (basic and derived), line count, branch count, misc, error count and density metrics in [11] [19]. Another way SVM was used to predict defects was in [14] using log files. SVM was applied on context vectors obtained with random indexing which is a text analysis technique. The context vectors represent an entry in a log file which represents an event in the system. [18] held a comparative study between the performance of SVM and ANN, The result of the research showed that SVM outperformed ANN when viewing fault-proneness prediction as a binary classification task with the NASA dataset.

In the work of [55] [7] [27] , RF models were applied for DP tasks. RF model is an ML model that consists of a set of decision trees that classify input data. The

output of RF is chosen from the majority predicted value from decision trees. [20] and [34] studied the effectiveness of random forests comparing to some other classification models (logistic regression, Naive Bayes, Discriminant analysis) on NASA datasets. On the same dataset, [11] showed that SVM, RF, and MLP outperformed KNN, Logistic Regression, RBF, and DT. For CM1dataset, SVM slightly outperformed RF and MLP, RF outperformed SVM and MLP for PC1 dataset and MLP outperformed RF and SVM for KC1 and KC3 dataset. In [7], a comparative study of the performance of four classifiers SVM, RF, and NB on NASA datasets. RF got the best result, however, when measuring the stability of the model for the DP task, NB showed more stability compared to other classifiers, which revealed that the model had a low level of flipping in different runs which in opposite SVM is the less stable in KC4 dataset due to its sensitivity to the training data.

Causal methods came to offer better decision making compared to regression models. BN is a directed graph where the nodes represent uncertain variables and arcs that represent a causal relationship between variables [13] [44]. According to [13] the power of BN is its flexibility since the probability values can be assigned to variables and hence the joint probability distribution can be recalculated and conditioned using Bayesian propagation. However, Bayesian models suffer from subjectivity when introducing expert judgements, but their main advantages are using a limited number of metrics and coping with missing data. In [7] NB showed more stability in predicting defects on many datasets compared to SVM and RF.

5.2 Deep Learning Methods

Recent studies showed that neural networks are being applied for classifying modules into defect-prone and non-defect-prone. Feedforward neural network allows a model to learn by propagating the adjusted weights on neurons calculated from the output. The input layer inputs the calculated software metrics which are fed forward into the neural network and the output layer outputs the result of the classification of the module. Through the experiments of [11] [15] the accuracy obtained with neural networks in overall is as good as other ML models like SVM and RF.

CNN provided a new approach for DP tasks than other ML models built using hand-crafted features [35] that correlate with defects but don't capture the semantic of the code source [9]. The result of their work showed an improvement of 12% and 16% in terms of F-measure, comparing to DBN-based and traditional features-based respectively. In [35] CNN was used for DP task as a feature generator to generate semantic and structural features of the source code which was later combined with logistic regression as classifier.

Software predictions using deep generative models were investigated by [22]. The result of their work shows that Stack Sparse Auto-Encoder (SSAE) and Deep Belief Network (DBN) achieve better performance. In the work of [5] , different types of neural networks have been used for DP, which are namely Feedforward Neural Network (FNN), Recurrent Neural Network (RNN), Artificial Neural Net-

work (ANN), and Deep Neural Network(DNN). In the experiment Genetic Algorithm (GA) was used to select the features and Particle Swarm Optimization (PSO) for building the cluster. The models were trained on different datasets from NASA promise software engineering repository. The result of the study reveals that DNN gave the best accuracy.

6 Discussion

The overview of the literature shows that in DP, statistical techniques are used in the pre-modeling step and model evaluation, while machine learning models are utilized for the predictions. Descriptive statistics are used to understand the distribution of the data of each variable or samples and identify the possibility of outliers. Inferential statistics helps in identifying the correlation between variables, normalizing data, proceed hypothesis testing for identifying the relevant features for a model. However, sometimes the features are not related directly to defect so that their relation is not significant statistically. Moreover, a machine learning algorithm can cover this problem so that it can predict the defect more accurately.

Many papers studied the performance of ML models on specific datasets such as SVM, Random Forests, Bayes Networks, MLP, NN, CNN, and LSTM. When outliers are present in the dataset SVM's tend to be more effective since they can control them using the margin parameter C, but also they are good when dealing with higher dimensional data, the density functions allow to map to higher dimensions in an optimized way [11]. Random Forest is too robust to outliers and noisy data which make their accuracy high and more significant for large datasets [20]. Logistic regression is also as good as many ML models when features and outputs are linearly related but it doesn't allow to combine multiple features at the same time [56]. Therefore, techniques of multiple classifications as one vs one and one vs all must be used for multiple classifications. However, these machine learning models suffer from different limitations that are hard to reach a consensus about which model is better. We remark many researchers claiming that some models are better than others for some specific datasets and specific metrics. The generalization of results is still questionable till today due to many factors that influence experiments such as metrics chosen and datasets, because the performance of a model is highly dependent on the dataset and metrics [44]. [16] study affirms that classifiers perform differently in different datasets and the focus should be on studying different classifiers instead of proving the performance of one. Ensemble learning in this case can be adequate to shift the focus from the model to the quality of dataset and metrics to use.

Moreover, there here has been little reasoning about the choice and size of software metrics. Appropriate features are very important for the model, it's essential to have a set of effective metrics that truly contribute to DP to reduce the parameter size of the model, but also to understand the relationship between them and their effect on the prediction. The work of [44] highlighted the relationship between

different software metrics and defects tested on different datasets. It would be interesting to explore the effectiveness of obtained metrics, first, in predicting defects and, second, on datasets used in the experiments adding the NASA datasets to clarify assumptions made about the influence of metrics, dataset, and the model. We also recommend the use of techniques of feature selection like cross-validation or inferential statistic techniques before conducting any experiment.

The NASA datasets are the most used by researchers in this area, it allows comparing the performance of methods proposed in the literature, but they do not help to generalize results to other software. Moreover, NASA datasets suffer from duplicated cases [50] especially in MC1, PC2 and PC5 datasets the rate of repeated instances are estimated 79%, 75% and 90% respectively [19] . In ML duplicated cases prevent the model from learning useful patterns since data that are in training are seen in the validation. Also, reducing the size of the dataset can bias the model and hence less accurate results. For this reason, data pre-processing should be conducted properly before any experiment.

Another problem that is encountered in DP is the imbalanced data, which means that samples of some classes are frequent than others. This phenomenon influences the classifiers generally by over-predicting majority classes. The survey [30] shows that using the resampling technique can help to overcome this data imbalance problem.

7 Conclusion and Future Research

Software bugs, errors, defects, or different names can be assigned to them, but they remain a quality problem in software engineering. Experts claim that 3 or 4 defects per 1000 lines of code are enough to make a drastic degradation of the quality of the software. Moreover, the impact of defects are negative on economic, business and human sides [47]. Defect prediction aims to automate the prediction of defects and help testers in spotting and correcting the right defective components for reliable software.

Throughout this paper, we surveyed the application of Data Science in DP, whose contribution to this domain is remarkable. The literature is abundant of works that interest in this area of software engineering. Among the sub-fields of Data science, ML models got the largest part of the interest of researchers. Many ML models have been applied to different datasets with different techniques, and many empirical studies were conducted to measure the effectiveness and power of models against others. The inferential and descriptive statistic concepts were not addressed specifically in DP. However, they were used and present in most of the experiments to measure the goodness of fit and the goodness of estimate of models, and in other cases to evaluate the capability of models in different conditions.

References

1. Ieee standard glossary of software engineering terminology. IEEE Std 610.12-1990 pp. 1–84 (1990). https://doi.org/10.1109/IEEESTD.1990.101064
2. Agarwal, S., Gupta, S., Aggarwal, R., Maheshwari, S., Goel, L., Gupta, S.: Substantiation of software defect prediction using statistical learning: An empirical study. In: 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU). pp. 1–6. IEEE (2019)
3. Akinsanya, B., Araújo, L.J., Charikova, M., Gimaeva, S., Grichshenko, A., Khan, A., Mazzara, M., Shilintsev, D., et al.: Machine learning and value generation in software development: a survey. Proceeding of International Conference on Software Testing, Machine Learning and Complex Process Analysis (TMPA-2019) (2019)
4. Alsawalqah, H., Faris, H., Aljarah, I., Alnemer, L., Alhindawi, N.: Hybrid smote-ensemble approach for software defect prediction. In: Computer Science On-line Conference. pp. 355–366. Springer (2017)
5. Ayon, S.I.: Neural network based software defect prediction using genetic algorithm and particle swarm optimization. In: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). pp. 1–4. IEEE (2019)
6. Basili, V.R., Briand, L.C., Melo, W.L.: A validation of object-oriented design metrics as quality indicators. IEEE Transactions on software engineering **22**(10), 751–761 (1996)
7. Bowes, D., Hall, T., Petrić, J.: Software defect prediction: do different classifiers find the same defects? Software Quality Journal **26**(2), 525–552 (2018)
8. Burnstein, I.: Practical software testing: a process-oriented approach. Springer Science & Business Media (2006)
9. Dam, H.K., Pham, T., Ng, S.W., Tran, T., Grundy, J., Ghose, A., Kim, T., Kim, C.J.: A deep tree-based model for software defect prediction. arXiv preprint arXiv:1802.00921 (2018)
10. Dam, H.K., Tran, T., Pham, T.T.M., Ng, S.W., Grundy, J., Ghose, A.: Automatic feature learning for predicting vulnerable software components. IEEE Transactions on Software Engineering (2018)
11. Elish, K.O., Elish, M.O.: Predicting defect-prone software modules using support vector machines. Journal of Systems and Software **81**(5), 649–660 (2008)
12. Evanco, W.M.: Poisson analyses of defects for small software components. Journal of Systems and Software **38**(1), 27–35 (1997)
13. Fenton, N., Neil, M., Marsh, W., Hearty, P., Marquez, D., Krause, P., Mishra, R.: Predicting software defects in varying development lifecycles using bayesian nets. Information and Software Technology **49**(1), 32–43 (2007)
14. Fronza, I., Sillitti, A., Succi, G., Terho, M., Vlasenko, J.: Failure prediction based on log files using random indexing and support vector machines. Journal of Systems and Software **86**(1), 2–11 (2013)
15. Gayathri, M., Sudha, A.: Software defect prediction system using multilayer perceptron neural network with data mining. International Journal of Recent Technology and Engineering **3**(2), 54–59 (2014)
16. Ghotra, B., McIntosh, S., Hassan, A.E.: Revisiting the impact of classification techniques on the performance of defect prediction models. In: Proceedings of the 37th International Conference on Software Engineering-Volume 1. pp. 789–800. IEEE Press (2015)
17. Gilb, T., Graham, D.: Software inspections. Addison-Wesley Reading, Masachusetts (1993)
18. Gondra, I.: Applying machine learning to software fault-proneness prediction. Journal of Systems and Software **81**(2), 186–195 (2008)
19. Gray, D., Bowes, D., Davey, N., Sun, Y., Christianson, B.: Using the support vector machine as a classification method for software defect prediction with static code metrics. In: International Conference on Engineering Applications of Neural Networks. pp. 223–234. Springer (2009)
20. Guo, L., Ma, Y., Cukic, B., Singh, H.: Robust prediction of fault-proneness by random forests. In: 15th international symposium on software reliability engineering. pp. 417–428. IEEE (2004)

21. Gyimothy, T., Ferenc, R., Siket, I.: Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Transactions on Software engineering **31**(10), 897–910 (2005)
22. Hasanpour, A., Farzi, P., Tehrani, A., Akbari, R.: Software defect prediction based on deep learning models: Performance study. arXiv preprint arXiv:2004.02589 (2020)
23. Holschuh, T., Pauser, M., Herzig, K., Zimmermann, T., Premraj, R., Zeller, A.: Predicting defects in sap java code: An experience report. In: 2009 31st International Conference on Software Engineering-Companion Volume. pp. 172–181. IEEE (2009)
24. Huang, F., Strigini, L.: Predicting software defects based on cognitive error theories. In: 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). pp. 134–135. IEEE (2018)
25. Humphrey, W.S.: A discipline for software engineering. Pearson Education India (1995)
26. Igual, L., Seguí, S.: Introduction to data science. In: Introduction to Data Science, pp. 1–4. Springer (2017)
27. Jacob, S.G., et al.: Improved random forest algorithm for software defect prediction through data mining techniques. International Journal of Computer Applications **117**(23) (2015)
28. Janes, A., Scotto, M., Pedrycz, W., Russo, B., Stefanovic, M., Succi, G.: Identification of defect-prone classes in telecommunication software systems using design metrics. Information sciences **176**(24), 3711–3734 (2006). https://doi.org/10.1016/j.ins.2005.12.002, http://dx.doi.org/10.1016/j.ins.2005.12.002
29. Ji, H., Huang, S., Lv, X., Wu, Y., Feng, Y.: Empirical studies of a kernel density estimation based naive bayes method for software defect prediction. IEICE TRANSACTIONS on Information and Systems **102**(1), 75–84 (2019)
30. Kamei, Y., Shihab, E., Adams, B., Hassan, A.E., Mockus, A., Sinha, A., Ubayashi, N.: A large-scale empirical study of just-in-time quality assurance. IEEE Transactions on Software Engineering **39**(6), 757–773 (2012)
31. Kampenes, V.B., Dybå, T., Hannay, J.E., Sjøberg, D.I.: A systematic review of effect size in software engineering experiments. Information and Software Technology **49**(11-12), 1073–1086 (2007)
32. Laitenberger, O.: Studying the effects of code inspection and structural testing on software quality. In: Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No. 98TB100257). pp. 237–246. IEEE (1998)
33. Laitenberger, O., DeBaud, J.M.: An encompassing life cycle centric survey of software inspection. Journal of systems and software **50**(1), 5–31 (2000)
34. Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering **34**(4), 485–496 (2008)
35. Li, J., He, P., Zhu, J., Lyu, M.R.: Software defect prediction via convolutional neural network. In: 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS). pp. 318–328. IEEE (2017)
36. Li, L., Lessmann, S., Baesens, B.: Evaluating software defect prediction performance: an updated benchmarking study. arXiv preprint arXiv:1901.01726 (2019)
37. Liu, H.H.: Software performance and scalability: a quantitative approach, vol. 7. John Wiley & Sons (2011)
38. Madeyski, L., Jureczko, M.: Which process metrics can significantly improve defect prediction models? an empirical study. Software Quality Journal **23**(3), 393–422 (2015)
39. Malhotra, R.: Comparative analysis of statistical and machine learning methods for predicting faulty modules. Applied Soft Computing **21**, 286–297 (2014)
40. Malhotra, R., Jain, A.: Fault prediction using statistical and machine learning methods for improving software quality. Journal of Information Processing Systems **8**(2), 241–262 (2012)
41. Malhotra, R., Kaur, A., Singh, Y.: Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines. International Journal of System Assurance Engineering and Management **1**(3), 269–281 (2010)
42. Mäntylä, M.V., Lassenius, C.: What types of defects are really discovered in code reviews? IEEE Transactions on Software Engineering **35**(3), 430–448 (2008)

43. McGraw, G.: Software security. IEEE Security & Privacy **2**(2), 80–83 (2004)
44. Okutan, A., Yıldız, O.T.: Software defect prediction using bayesian networks. Empirical Software Engineering **19**(1), 154–181 (2014)
45. Panichella, A., Oliveto, R., De Lucia, A.: Cross-project defect prediction models: L'union fait la force. In: 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE). pp. 164–173. IEEE (2014)
46. Pobereżnik, Ł.: A method for selecting environments for software compatibility testing. In: 2013 Federated Conference on Computer Science and Information Systems. pp. 1355–1360. IEEE (2013)
47. Pressman, R.S.: Software engineering: a practitioner's approach. Palgrave Macmillan (2005)
48. Runeson, P., Andersson, C., Thelin, T., Andrews, A., Berling, T.: What do we know about defect detection methods?[software testing]. IEEE software **23**(3), 82–90 (2006)
49. Runeson, P., Wohlin, C.: An experimental evaluation of an experience-based capture-recapture method in software code inspections. Empirical Software Engineering **3**(4), 381–406 (1998)
50. Shepperd, M., Song, Q., Sun, Z., Mair, C.: Data quality: Some comments on the nasa software defect datasets. IEEE Transactions on Software Engineering **39**(9), 1208–1215 (2013)
51. Singh, Y., Kaur, A., Malhotra, R.: Empirical validation of object-oriented metrics for predicting fault proneness models. Software quality journal **18**(1), 3 (2010)
52. Sullivan, M., Chillarege, R.: Software defects and their impact on system availability: A study of field failures in operating systems. In: FTCS. vol. 21, pp. 2–9 (1991)
53. Thapaliyal, M., Verma, G.: Software defects and object oriented metrics-an empirical analysis. International Journal of Computer Applications **9**(5), 41–44 (2010)
54. Wiegers, K.E.: Peer reviews in software: A practical guide. Addison-Wesley Boston (2002)
55. Yang, X., Lo, D., Xia, X., Sun, J.: Tlel: A two-layer ensemble learning approach for just-in-time defect prediction. Information and Software Technology **87**, 206–220 (2017)
56. Yang, X., Lo, D., Xia, X., Zhang, Y., Sun, J.: Deep learning for just-in-time defect prediction. In: 2015 IEEE International Conference on Software Quality, Reliability and Security. pp. 17–26. IEEE (2015)
57. Zhou, Y., Leung, H.: Empirical analysis of object-oriented design metrics for predicting high and low severity faults. IEEE Transactions on software engineering **32**(10), 771–789 (2006)
58. Zimmermann, T., Nagappan, N.: Predicting defects using network analysis on dependency graphs. In: 2008 ACM/IEEE 30th International Conference on Software Engineering. pp. 531–540. IEEE (2008)