Bibliography

[1] About debian: https://www.debian.org/.

[2] About eclipse: https://en.wikipedia.org/wiki/eclipse-(software).

[3] About mozilla: https://www.mozilla.org/en-us/firefox/.

[4] Adf test: Microsoft excel add-in.

[5] Argouml database: argouml.tigris.org/source/browse/argouml/trunk/src/.

[6] Eclipse bug repository: https://bugs.eclipse.org/bugs/.

[7] Eclipse metrics plugin: metrics.sourceforge.net/.

[8] Function approximation and nonlinear regression: http://www.mathworks.in/help/nnet/function-approximation-and-nonlinear-regression.html.

[9] K-median-clustering: https://www.cs.princeton.edu/courses/archive/fall14/cos521/projects/kmedian

[10] Markov models, hidden and otherwise: http://kochanski.org/gpk/teaching/0401oxford.

[11] Mozilla bug repository: https://bugzilla.mozilla.org/.

[12] Nonparametric estimation: http://mathworld.wolfram.com/nonparametricestimation.html.

[13] Open source software directory: http://www.ohloh.net/.

[14] Software fault prediction using fuzzy clustering and radial-basis function network.

[15] Ultimate debian database(bug repository): http://udd.debian.org/.

[16] Abd-El-Hafiz, S. K. (2011). Efficient detection of function clones in software systems using the fractal dimension and metrics. In *Int. Conf. Software Engineering*, pages 88–94.

[17] Abd-El-Hafiz, S. K. (2012). A metrics-based data mining approach for software clone detection. In *2012 IEEE 36th Annual Computer Software and Applications Conference*, pages 35–41. IEEE.

[18] Alan, O. and Catal, P. D. C. (2009). An outlier detection algorithm based on object-oriented metrics thresholds. In *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*, pages 567–570. IEEE.

[19] Aljahdali, S. H. and El-Telbany, M. E. (2009). Software reliability prediction using multi-objective genetic algorithm. In *2009 IEEE/ACS International Conference on Computer Systems and Applications*, pages 293–300. IEEE.

[20] Amin, A., Grunske, L., and Colman, A. (2013). An approach to software reliability prediction based on time series modeling. *Journal of Systems and Software*, 86(7):1923–1932.

[21] An, L. and Khomh, F. (2015). An empirical study of highly impactful bugs in mozilla projects. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*, pages 262–271. IEEE.

[22] Antoniol, G., Penta, M. D., Casazza, G., and Merlo, E. (2001). Modeling clones evolution through time series. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01)*, page 273. IEEE Computer Society.

[23] April, A. and Abran, A. (2012). *Software maintenance management: evaluation and continuous improvement*, volume 67. John Wiley & Sons.

[24] Areekul, P., Senjyu, T., Toyama, H., and Yona, A. (2009). Combination of artificial neural network and arima time series models for short term price forecasting in deregulated market. In *Transmission & Distribution Conference & Exposition: Asia and Pacific, 2009*, pages 1–4. IEEE.

[25] Bai, C., Hu, Q., Xie, M., and Ng, S. H. (2005). Software failure prediction based on a markov bayesian network model. *Journal of Systems and Software*, 74(3):275–282.

[26] Barghout, M., Littlewood, B., and Abdel-Ghaly, A. (1997). A non-parametric approach to software reliability prediction. In *Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on*, pages 366–377. IEEE.

[27] Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M., and Bier, L. (1998). Clone detection using abstract syntax trees. In *Software Maintenance, 1998. Proceedings., International Conference on*, pages 368–377. IEEE.

[28] Bellon, S., Koschke, R., Antoniol, G., Krinke, J., and Merlo, E. (2007). Comparison and evaluation of clone detection tools. *IEEE Transactions on Software Engineering*, 33(9):577–591.

[Bhatnagar and Kakkar] Bhatnagar, R. and Kakkar, M. Predicting software reliability using machine learning approach for sdlc life cycle.

[30] Bibi, S., Tsoumakas, G., Stamelos, I., and Vlahavas, I. (2008). Regression via classification applied on software defect estimation. *Expert Systems with Applications*, 34(3):2091–2101.

[31] Binkley, D., Feild, H., Lawrie, D., and Pighin, M. (2007). Software fault prediction using language processing. In *Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, 2007. TAICPART-MUTATION 2007*, pages 99–110. IEEE.

[32] Bishnu, P. S. and Bhattacherjee, V. (2012). Software fault prediction using quad tree-based k-means clustering algorithm. *IEEE Transactions on knowledge and data engineering*, 24(6):1146–1150.

[33] Boetticher, G. D. (2006). Improving credibility of machine learner models in software engineering. *Advanced Machine Learner Applications in Software Engineering (Series on Software Engineering and Knowledge Engineering)*, pages 52–72.

[34] Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

[35] Brockwell, P. J. and Davis, R. A. (2013). *Time series: theory and methods*. Springer Science & Business Media.

[36] Calefato, F., Lanubile, F., and Mallardo, T. (2004). Function clone detection in web applications: a semiautomated approach. *Journal of Web Engineering*, 3:3–21.

[37] Carlson, D., Haynsworth, E., and Markham, T. (1974). A generalization of the schur complement by means of the moore–penrose inverse. *SIAM Journal on Applied Mathematics*, 26(1):169–175.

[38] Castillo, E., Guijarro-Berdiñas, B., Fontenla-Romero, O., and Alonso-Betanzos, A. (2006). A very fast learning method for neural networks based on sensitivity analysis. *J. Mach. Learn. Res.*, 7:1159–1182.

[39] Chakraborty, K., Mehrotra, K., Mohan, C. K., and Ranka, S. (1992). Forecasting the behavior of multivariate time series using neural networks. *Neural networks*, 5(6):961–970.

[40] Challagulla, V. U., Bastani, F. B., and Yen, I.-L. (2006). A unified framework for defect data analysis using the mbr technique. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 39–46. IEEE.

[41] Challagulla, V. U., Bastani, F. B., Yen, I.-L., and Paul, R. A. (2005). Empirical assessment of machine learning based software defect prediction techniques. In *Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on*, pages 263–270. IEEE.

[42] Chang, C.-P., Chu, C.-P., and Yeh, Y.-F. (2009). Integrating in-process software defect prediction with association mining to discover defect pattern. *Information and Software Technology*, 51(2):375–384.

[43] Chen, J., Liu, S., Chen, X., Gu, Q., and Chen, D. (2013). Empirical studies on feature selection for software fault prediction. In *Proceedings of the 5th Asia-Pacific Symposium on Internetware*, page 26. ACM.

[44] Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on software engineering*, 20(6):476–493.

[45] Chow, T. S. (1978). Testing software design modeled by finite-state machines. *IEEE transactions on software engineering*, (3):178–187.

[46] Cockburn, A. (2002). *Agile software development*, volume 177. Addison-Wesley Boston.

[47] Coello, C. C., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media.

[48] Cukic, B. and Ma, Y. (2007). Predicting fault-proneness: Do we finally know how. *Reliability analysis of system failure data, Cambridge, UK*.

[49] das Chagas Moura, M., Zio, E., Lins, I. D., and Droguett, E. (2011). Failure and reliability prediction by support vector machines regression of time series data. *Reliability Engineering & System Safety*, 96(11):1527–1534.

[50] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[51] Dejaeger, K., Verbraken, T., and Baesens, B. (2013). Toward comprehensible software fault prediction models using bayesian network classifiers. *IEEE Transactions on Software Engineering*, 39(2):237–257.

[52] Denaro, G., Lavazza, L., and Pezze, M. (2003). An empirical evaluation of object oriented metrics in industrial setting. In *The 5th CaberNet Plenary Workshop, Porto Santo, Madeira Archipelago, Portugal*.

[53] Di Lucca, G. A., Di Penta, M., and Fasolino, A. R. (2002). An approach to identify duplicated web pages. In *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*, pages 481–486. IEEE.

[54] Ducasse, S., Rieger, M., and Demeyer, S. (1999). A language independent approach for detecting duplicated code. In *Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on*, pages 109–118. IEEE.

[55] D'Ambros, M., Lanza, M., and Robbes, R. (2012). Evaluating defect prediction approaches: a benchmark and an extensive comparison. *Empirical Software Engineering*, 17(4-5):531–577.

[56] El Emam, K., Melo, W., and Machado, J. C. (2001). The prediction of faulty classes using object-oriented design metrics. *Journal of Systems and Software*, 56(1):63–75.

[57] Elish, M. O., Al-Yafei, A. H., and Al-Mulhem, M. (2011). Empirical comparison of three metrics suites for fault prediction in packages of object-oriented systems: A case study of eclipse. *Advances in Engineering Software*, 42(10):852–859.

[58] Eswaran, C., Logeswaran, R., et al. (2012). An enhanced hybrid method for time series prediction using linear and neural network models. *Applied Intelligence*, 37(4):511–519.

[59] Fachinotti, V., Anca, A., and Cardona, A. (2011). A method for the solution of certain problems in least squares. *Int J Numer Method Biomed Eng*, 27(4):595–607.

[60] Fathima, T. and Jothiprakash, V. (2014). Behavioural analysis of a time series–a chaotic approach. *Sadhana*, 39(3):659–676.

[61] Fenton, N., Neil, M., and Marquez, D. (2008). Using bayesian networks to predict software defects and reliability. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 222(4):701–712.

[62] Fenton, N. E. and Neil, M. (1999). A critique of software defect prediction models. *IEEE Transactions on software engineering*, 25(5):675–689.

[63] Fung, K., Kwong, C., Siu, K. W., and Yu, K. (2012). A multi-objective genetic algorithm approach to rule mining for affective product design. *Expert Systems with Applications*, 39(8):7411–7419.

[64] Gao, K. and Khoshgoftaar, T. M. (2007). A comprehensive empirical study of count models for software fault prediction. *IEEE Transactions on Reliability*, 56(2):223–236.

[65] Ghosh, A. and Chakraborty, M. (2012). Hybrid optimized back propagation learning algorithm for multi-layer perceptron. *International Journal of Computer Applications*, 60(13).

[66] Göde, N. and Koschke, R. (2009). Incremental clone detection. In *Software Maintenance and Reengineering, 2009. CSMR'09. 13th European Conference on*, pages 219–228. IEEE.

[67] Goel, A. L. (1985). Software reliability models: Assumptions, limitations, and applicability. *IEEE Transactions on software engineering*, (12):1411–1423.

[68] Goel, A. L. and Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE transactions on Reliability*, 3:206–211.

[69] Grubb, P. and Takang, A. A. (2003). *Software maintenance: concepts and practice*. World Scientific.

[70] Guo, L., Cukic, B., and Singh, H. (2003). Predicting fault prone modules by the dempster-shafer belief networks. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, pages 249–252. IEEE.

[71] Guo, P. and Lyu, M. R. (2000). Software quality prediction using mixture models with em algorithm. In *Quality Software, 2000. Proceedings. First Asia-Pacific Conference on*, pages 69–78. IEEE.

[72] Gyimothy, T., Ferenc, R., and Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software engineering*, 31(10):897–910.

[Hall et al.] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. The weka data mining software: an update. acm sigkdd explor newslett 2009; 11: 10–8.

[74] Hamilton, J. D. (1994). *Time series analysis*, volume 2. Princeton university press Princeton.

[75] Hassan, A. E. and Holt, R. C. (2005). The top ten list: Dynamic fault prediction. In *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, pages 263–272. IEEE.

[76] Herraiz, I., Gonzalez-Barahona, J. M., and Robles, G. (2007). Forecasting the number of changes in eclipse using time series analysis. In *Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on*, pages 32–32. IEEE.

[77] Hu, S. (2007). Akaike information criterion. *Center for Research in Scientific Computation*, 93.

[78] Huang, C.-Y. and Lyu, M. R. (2011). Estimation and analysis of some generalized multiple change-point software reliability models. *IEEE Transactions on reliability*, 60(2):498–514.

[79] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE.

[80] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501.

[81] Illes-Seifert, T. and Paech, B. (2008). Exploring the relationship of a file's history and its fault-proneness: An empirical study. In *Practice and Research Techniques, 2008. TAIC PART'08. Testing: Academic & Industrial Conference*, pages 13–22. IEEE.

[82] Im, K. S., Pesaran, M. H., and Shin, Y. (2003). Testing for unit roots in heterogeneous panels. *Journal of econometrics*, 115(1):53–74.

[83] Irrera, I., Duraes, J., and Vieira, M. (2014). On the need for training failure prediction algorithms in evolving software systems. In *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering*, pages 216–223. IEEE.

[84] ISO, I. (2010). Iec/ieee 24765: 2010 systems and software engineering-vocabulary. Technical report, Technical report, Institute of Electrical and Electronics Engineers, Inc.

[85] Jiang, L., Misherghi, G., Su, Z., and Glondu, S. (2007a). Deckard: Scalable and accurate tree-based detection of code clones. In *Proceedings of the 29th international conference on Software Engineering*, pages 96–105. IEEE Computer Society.

[86] Jiang, Y., Cukic, B., and Menzies, T. (2007b). Fault prediction using early lifecycle data. In *Software Reliability, 2007. ISSRE'07. The 18th IEEE International Symposium on*, pages 237–246. IEEE.

[87] Johnson, J. H. (1994). Visualizing textual redundancy in legacy source. In *Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research*, page 32. IBM Press.

[88] Junhong, G., Xiaozong, Y., and Hongwei, L. (2005). Software reliability nonlinear modeling and its fuzzy evaluation. In *4th WSEAS International Conference*.

[89] Kamiya, T., Kusumoto, S., and Inoue, K. (2002). Ccfinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering*, 28(7):654–670.

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

[91] Kapser, C. and Godfrey, M. W. (2006). " cloning considered harmful" considered harmful. In *Reverse Engineering, 2006. WCRE'06. 13th Working Conference on*, pages 19–28. IEEE.

[92] Kapur, P., Pham, H., Anand, S., and Yadav, K. (2011). A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation. *IEEE Transactions on Reliability*, 60(1):331–340.

[93] Karunanithi, N., Whitley, D., and Malaiya, Y. K. (1992). Using neural networks in reliability prediction. *IEEE Software*, 9(4):53–59.

[94] Kaur, J., Kumar, R., and Kaur, S. (2017). Design code clone detection system uses optimal and intelligence technique based on software engineering. *International Journal of Advanced Research in Computer Science*, 8(5).

[95] Khashei, M. and Bijari, M. (2010). An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Systems with applications*, 37(1):479–489.

[96] Khashei, M. and Bijari, M. (2011). A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, 11(2):2664–2675.

[97] Khoshgoftaar, T. M., Allen, E. B., and Busboom, J. C. (2000). Modeling software quality: the software measurement analysis and reliability toolkit. In *Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. 12th IEEE International Conference on*, pages 54–61. IEEE.

[98] Khoshgoftaar, T. M., Gao, K., and Szabo, R. M. (2001). An application of zero-inflated poisson regression for software fault prediction. In *Software Reliability Engineering, 2001. ISSRE 2001. Proceedings. 12th International Symposium on*, pages 66–73. IEEE.

[99] Khoshgoftaar, T. M., Geleyn, E., and Gao, K. (2002). An empirical study of the impact of count models predictions on module-order models. In *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, pages 161–172. IEEE.

[100] Khoshgoftaar, T. M. and Seliya, N. (2002). Tree-based software quality estimation models for fault prediction. In *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, pages 203–214. IEEE.

[101] Khoshgoftaar, T. M. and Seliya, N. (2003a). Fault prediction modeling for software quality estimation: Comparing commonly used techniques. *Empirical Software Engineering*, 8(3):255–283.

[102] Khoshgoftaar, T. M. and Seliya, N. (2003b). Software quality classification modeling using the sprint decision tree algorithm. *International Journal on Artificial Intelligence Tools*, 12(03):207–225.

[103] Khoshgoftaar, T. M. and Seliya, N. (2004). Comparative assessment of software quality classification techniques: An empirical case study. *Empirical Software Engineering*, 9(3):229–257.

[104] Khoshgoftaar, T. M., Seliya, N., and Gao, K. (2005). Assessment of a new three-group software quality classification technique: An empirical case study. *Empirical Software Engineering*, 10(2):183–218.

[105] Khoshgoftaar, T. M., Seliya, N., and Sundaresh, N. (2006). An empirical study of predicting software faults with case-based reasoning. *Software Quality Journal*, 14(2):85–111.

[106] Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011a). Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks*, 22(9):1341–1356.

[107] Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011b). Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE Transactions on Neural Networks*, 22(3):337–346.

[108] Kiran, N. R. and Ravi, V. (2008). Software reliability prediction by soft computing techniques. *Journal of Systems and Software*, 81(4):576–583.

[109] Koch, S. (2004). Agile principles and open source software development: A theoretical and empirical. In *Extreme Programming and Agile Processes in Software Engineering: 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany, June 6-10, 2004, Proceedings*, volume 3092, page 85. Springer.

[110] Komondoor, R. and Horwitz, S. (2001). Using slicing to identify duplication in source code. In *International Static Analysis Symposium*, pages 40–56. Springer.

[111] Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007.

[112] Koru, A. G. and Liu, H. (2005a). Building effective defect-prediction models in practice. *IEEE software*, 22(6):23–29.

[113] Koru, A. G. and Liu, H. (2005b). An investigation of the effect of module size on defect prediction using static measures. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 1–5. ACM.

[114] Koru, A. G. and Liu, H. (2007). Identifying and characterizing change-prone classes in two large-scale open-source products. *Journal of Systems and Software*, 80(1):63–73.

[115] Koru, A. G. and Tian, J. (2003). An empirical comparison and characterization of high defect and high complexity modules. *Journal of Systems and Software*, 67(3):153–163.

[116] Koschke, R., Falke, R., and Frenzel, P. (2006). Clone detection using abstract syntax suffix trees. In *2006 13th Working Conference on Reverse Engineering*, pages 253–262. IEEE.

[117] Krinke, J. (2001). Identifying similar code with program dependence graphs. In *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*, pages 301–309. IEEE.

[118] Kumar, G. and Bhatia, P. K. (2014). Comparative analysis of software engineering models from traditional to modern methodologies. In *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, pages 189–196. IEEE.

[119] Lakshmanan, I. and Ramasamy, S. (2015). An artificial neural-network approach to software reliability growth modeling. *Procedia Computer Science*, 57:695–702.

[120] Larose, D. T. (2005). K-nearest neighbor algorithm. *Discovering Knowledge in Data: An Introduction to Data Mining*, pages 90–106.

[121] Lee, S. and Jeong, I. (2005). Sdd: high performance code clone detection system for large scale source code. In *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 140–141. ACM.

[122] Li, K., Liu, W., Zhao, K., Shao, M., and Liu, L. (2015). A novel dynamic weight neural network ensemble model. *International Journal of Distributed Sensor Networks*.

[123] Li, P. L., Herbsleb, J., Shaw, M., and Robinson, B. (2006). Experiences and results from initiating field defect prediction and product test prioritization efforts at abb inc. In *Proceedings of the 28th international conference on Software engineering*, pages 413–422. ACM.

[124] Li, Z. and Reformat, M. (2007). A practical method for the software fault-prediction. In *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*, pages 659–666. IEEE.

[125] Liu, C., Chen, C., Han, J., and Yu, P. S. (2006). Gplag: detection of software plagiarism by program dependence graph analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 872–881. ACM.

[126] Liu, H., Tian, H.-q., Pan, D.-f., and Li, Y.-f. (2013). Forecasting models for wind speed using wavelet, wavelet packet, time series and artificial neural networks. *Applied Energy*, 107:191–208.

[127] Liu, W., Liu, S., Gu, Q., Chen, X., and Chen, D. (2015). Fecs: A cluster based feature selection method for software fault prediction with noises. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, volume 2, pages 276–281. IEEE.

[128] Livieri, S., Higo, Y., Matushita, M., and Inoue, K. (2007). Very-large scale code clone analysis and visualization of open source programs using distributed ccfinder: D-ccfinder. In *29th International Conference on Software Engineering (ICSE'07)*, pages 106–115. IEEE.

[129] Lo, J.-H. (2009). The implementation of artificial neural networks applying to software reliability modeling. In *Control and Decision Conference, 2009. CCDC'09. Chinese*, pages 4349–4354. IEEE.

[130] Logofet, D. O. and Lesnaya, E. V. (2000). The mathematics of markov models: what markov chains can really predict in forest successions. *Ecological Modelling*, 126(2):285–298.

[131] Lyu, M. R. (2007). Software reliability engineering: A roadmap. In *2007 Future of Software Engineering*, pages 153–170. IEEE Computer Society.

[132] Lyu, M. R. et al. (1996). Handbook of software reliability engineering.

[133] Lyu, M. R. and Nikora, A. (1992). Applying reliability models more effectively (software). *IEEE software*, 9(4):43–52.

[134] Mahaweerawat, A., Sophatsathit, P., and Lursinsap, C. (2007). Adaptive self-organizing map clustering for software fault prediction. In *Fourth International Joint Conference on Computer Science and Software Engineering, KhonKaen, Thailand*, pages 35–41.

[135] Mahaweerawat, A., Sophatsathit, P., Lursinsap, C., and Musilek, P. (2004). Fault prediction in object-oriented software using neural network techniques. *Advanced Virtual and Intelligent Computing Center (AVIC), Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand*, pages 1–8.

[136] Manber, U. et al. (1994). Finding similar files in a large file system. In *Usenix Winter*, volume 94, pages 1–10.

[137] Marcus, A., Poshyvanyk, D., and Ferenc, R. (2008). Using the conceptual cohesion of classes for fault prediction in object-oriented systems. *IEEE Transactions on Software Engineering*, 34(2):287–300.

[138] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441.

[139] Mayrand, J., Leblanc, C., and Merlo, E. M. (1996). Experiment on the automatic detection of function clones in a software system using metrics. In *Software Maintenance 1996, Proceedings., International Conference on*, pages 244–253. IEEE.

[140] Mende, T. and Koschke, R. (2009). Revisiting the evaluation of defect prediction models. In *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, page 7. ACM.

[141] Menzies, T., DiStefano, J., Orrego, A., and Chapman, R. (2004). Assessing predictors of software defects. In *Proc. Workshop Predictive Software Models*.

[142] Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y., and Bener, A. (2010). Defect prediction from static code features: current results, limitations, new approaches. *Automated Software Engineering*, 17(4):375–407.

[143] Mertik, M., Lenic, M., Stiglic, G., and Kokol, P. (2006). Estimating software quality with advanced data mining techniques. In *Software Engineering Advances, International Conference on*, pages 19–19. IEEE.

[144] Milicic, D. (2005). Software quality models and philosophies. *Software quality attributes and trade-offs*, pages 3–19.

[145] Mohanthy, R., Naik, V., and Mubeen, A. (2014). Software reliability prediction by using ant colony optimization technique. In *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pages 496–500. IEEE.

[146] Moranda, P. and Jelinski, Z. (1972). Final report on software reliability study. *McDonnell Douglas Astronautics Company, MADC Report*, 11.

[147] Musa, J. (1979). Software reliability data, data analysis center for software. *Rome Air Development Center, Rome, NY*.

[148] Musa, J. D. (1975). A theory of software reliability and its application. *IEEE Transactions on Software Engineering*, (3):312–327.

[149] Musa, J. D., Iannino, A., and Okumoto, K. (1980). Software reliability measurement. *Journal of Systems and Software*, 1(3):223–241.

[150] Muthukumaran, K., Murthy, N. B., Reddy, G. K., and Talishetti, P. (2016). Testing and code review based effort-aware bug prediction model. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 17–30. Springer.

[151] Neamtiu, I., Xie, G., and Chen, J. (2013). Towards a better understanding of software evolution: an empirical study on open-source software. *Journal of Software: Evolution and Process*, 25(3):193–218.

[152] Nikora, A. P. and Munson, J. C. (2006). Building high-quality software fault predictors. *Software: Practice and Experience*, 36(9):949–969.

[153] Noekhah, S., Hozhabri, A. A., and Rizi, H. S. (2013). Software reliability prediction model based on ica algorithm and mlp neural network. In *e-Commerce in Developing Countries: With Focus on e-Security (ECDC), 2013 7th Intenational Conference on*, pages 1–15. IEEE.

[154] Nof, S. Y. (2009). *Springer handbook of automation*. Springer Science & Business Media.

[155] Olague, H. M., Etzkorn, L. H., Gholston, S., and Quattlebaum, S. (2007). Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. *IEEE Transactions on software Engineering*, 33(6).

[156] Ostrand, T. J., Weyuker, E. J., and Bell, R. M. (2005). Predicting the location and number of faults in large software systems. *IEEE Transactions on Software Engineering*, 31(4):340–355.

[157] Ostrand, T. J., Weyuker, E. J., and Bell, R. M. (2007). Automating algorithms for the identification of fault-prone files. In *Proceedings of the 2007 international symposium on Software testing and analysis*, pages 219–227. ACM.

[158] Pai, G. J. and Dugan, J. B. (2007). Empirical analysis of software fault content and fault proneness using bayesian methods. *IEEE Transactions on software Engineering*, 33(10).

[159] Pai, P.-F. and Hong, W.-C. (2006). Software reliability forecasting by support vector machines with simulated annealing algorithms. *Journal of Systems and Software*, 79(6):747–755.

[160] Palviainen, M., Evesti, A., and Ovaska, E. (2011). The reliability estimation, prediction and measuring of component-based software. *Journal of Systems and Software*, 84(6):1054–1070.

[161] Patenaude, J.-F., Merlo, E., Dagenais, M., and Laguë, B. (1999). Extending software quality assessment techniques to java systems. In *Program Comprehension, 1999. Proceedings. Seventh International Workshop on*, pages 49–56. IEEE.

[162] Pati, J., Kumar, B., Manjhi, D., and Shukla, K. (2017a). A comparison among arima, bp-nn and moga-nn forsoftware clone evolution prediction. *IEEE Access*.

[163] Pati, J., Kumar, B., Manjhi, D., and Shukla, K. (2017b). Machine learning strategies for temporal analysis of software clone evolution using software metrics. *International Journal of Applied Engineering Research*, 12(11):2798–2806.

[164] Pati, J. and Shukla, K. (2014a). A comparison of arima, neural network and a hybrid technique for debian bug number prediction. In *Computer and Communication Technology (ICCCT), 2014 International Conference on*, pages 47–53. IEEE.

[165] Pati, J. and Shukla, K. (2014b). A neural network approach to debian bug number prediction. In *Computer Application and Signal Processing*. CIIT.

[166] Pati, J. and Shukla, K. (2014c). Prediction of temporal bug patterns of debian using markov model. *nternational Journal of Advanced Research in Computer Science and Software Engineering*, 4(1):1–6.

[167] Pati, J. and Shukla, K. (2015a). A hybrid modeling approach for software clone evolution prediction. In *International Conference on Electrical Engineering and Computer Sciences*. HEF.

[168] Pati, J. and Shukla, K. (2015b). A nonlinear arima technique for debian bug number prediction. *International Journal of Software Engineering and Its Applications*, 4(4).

[169] Pati, J. and Shukla, K. (2017). Analysis of temporal bug patterns in open source software using hidden markov model. *International Journal of Software Engineering and Its Applications*, 11(4):11–24.

[170] Pati, J. and Shukla, K. K. (2015c). A hybrid technique for software reliability prediction. In *Proceedings of the 8th India Software Engineering Conference*, pages 139–146. ACM.

[171] Payal, A., Rai, C., and Reddy, B. (2013). Comparative analysis of bayesian regularization and levenberg-marquardt training algorithm for localization in wireless sensor network. In *Advanced Communication Technology (ICACT), 2013 15th International Conference on*, pages 191–194. IEEE.

[172] Pham, H. and Teng, X. (2006). Statistical models for predicting reliability of software systems in random environments. In *Springer Handbook of Engineering Statistics*, pages 507–520. Springer.

[173] Pigoski, T. M. (1996). *Practical software maintenance: best practices for managing your software investment*. Wiley Publishing.

[174] Pindyck, R. S. and Rubinfeld, D. L. (1998). *Econometric models and economic forecasts*, volume 4. Irwin/McGraw-Hill Boston.

[175] Pizzi, N. J., Summers, A. R., and Pedrycz, W. (2002). Software quality prediction using median-adjusted class labels. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2405–2409. IEEE.

[176] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[177] Rahman, F., Bird, C., and Devanbu, P. (2012). Clones: What is that smell? *Empirical Software Engineering*, 17(4-5):503–530.

[178] Raja, U., Hale, J. E., Hale, D. P., et al. (2011). Temporal patterns of software evolution defects: A comparative analysis of open source and closed source projects. *Journal of Software Engineering and Applications*, 4(08):497.

[179] Ramasamy, S. and Govindasamy, G. (2006). Generalized exponential poisson model for software reliability growth. *International Journal of Performability Engineering*, 2(3):291–301.

[180] Rathore, S. S. and Kumar, S. (2015). Predicting number of faults in software system using genetic programming. *Procedia Computer Science*, 62:303–311.

[181] Rattan, D., Bhatia, R., and Singh, M. (2013). Software clone detection: A systematic review. *Information and Software Technology*, 55(7):1165–1199.

[182] Ravi, V., Chauhan, N. J., and Kiran, N. R. (2009). Software reliability prediction using intelligent techniques: Application to operational risk prediction in firms. *International Journal of Computational Intelligence and Applications*, 8(02):181–194.

[183] Reformat, M. (2003). A fuzzy-based meta-model for reasoning about the number of software defects. In *International Fuzzy Systems Association World Congress*, pages 644–651. Springer.

[184] Riquelme, J., Ruiz, R., Rodríguez, D., and Moreno, J. (2008). Finding defective modules from highly unbalanced datasets. *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, 2(1):67–74.

[185] Robinson, D. and Dietrich, D. (1989). A nonparametric-bayes reliability-growth model. *IEEE Transactions on Reliability*, 38(5):591–598.

[186] Rossen, A. (2016). On the predictive content of nonlinear transformations of lagged autoregression residuals and time series observations. *Jahrbücher für Nationalökonomie und Statistik*, 236(3):389–409.

[187] Roy, C. K., Cordy, J. R., and Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, 74(7):470–495.

[188] Ruta, D. and Gabrys, B. (2007). Neural network ensembles for time series prediction. In *2007 International Joint Conference on Neural Networks*, pages 1204–1209. IEEE.

[189] Schneidewind, N. F. (2001). Investigation of logistic regression as a discriminant of software quality. In *Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*, pages 328–337. IEEE.

[190] Seliya, N. and Khoshgoftaar, T. M. (2007). Software quality estimation with limited fault data: a semi-supervised learning perspective. *Software Quality Journal*, 15(3):327–344.

[191] Shafi, S., Hassan, S. M., Arshaq, A., Khan, M. J., and Shamail, S. (2008). Software quality prediction techniques: A comparative analysis. In *Emerging Technologies, 2008. ICET 2008. 4th International Conference on*, pages 242–246. IEEE.

[192] Shanthini, A. and Chandrasekaran, R. (2014). Analyzing the effect of bagged ensemble approach for software fault prediction in class level and package level metrics. In *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pages 1–5. IEEE.

[193] Sharma, K., Garg, R., Nagpal, C., and Garg, R. (2010). Selection of optimal software reliability growth models using a distance based approach. *IEEE Transactions on Reliability*, 59(2):266–276.

[194] Shrestha, D. L. and Solomatine, D. P. (2006). Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2):225–235.

[195] Shumway, R. H. and Stoffer, D. S. (1982). An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264.

[196] Shumway, R. H. and Stoffer, D. S. (2006). Time series analysis and its application with r examples. *University of California, Davis, CA*.

[197] Shumway, R. H. and Stoffer, D. S. (2010). *Time series analysis and its applications: with R examples*. Springer Science & Business Media.

[198] Siegmund, J., Siegmund, N., and Apel, S. (2015). Views on internal and external validity in empirical software engineering. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 9–19. IEEE.

[199] Smith, R. and Horwitz, S. (2009). Detecting and measuring similarity in code clones. In *Proceedings of the International workshop on Software Clones (IWSC)*.

[200] Sneed, H. M. (2008). Offering software maintenance as an offshore service. In *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, pages 1–5. IEEE.

[201] Soleimani, A. and Asdaghi, F. (2014). An ais based feature selection method for software fault prediction. In *Intelligent Systems (ICIS), 2014 Iranian Conference on*, pages 1–5. IEEE.

[202] Stamp, M. (2004). A revealing introduction to hidden markov models. *Department of Computer Science San Jose State University*.

[203] Standard, I. (2006). Software engineering–software life cycle processes–maintenance. *ISO Standard*, 14764:2006.

[204] Subburaj, R., Gopal, G., and Kapur, P. (2007). A software reliability growth model for vital quality metrics. *South African Journal of Industrial Engineering*, 18(2):93.

[205] Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62.

[206] Thwin, M. M. T. and Quah, T.-S. (2005). Application of neural networks for software quality prediction using object-oriented metrics. *Journal of systems and software*, 76(2):147–156.

[207] Tomaszewski, P., Håkansson, J., Grahn, H., and Lundberg, L. (2007). Statistical models vs. expert estimation for fault prediction in modified code–an industrial case study. *Journal of Systems and Software*, 80(8):1227–1238.

[208] Tomaszewski, P., Lundberg, L., and Grahn, H. (2005). The accuracy of early fault prediction in modified code. In *Fifth Conference on Software Engineering Research and Practice in Sweden*, pages 57–63.

[209] Tosun, A., Turhan, B., and Bener, A. (2009). Validation of network measures as indicators of defective modules in software systems. In *Proceedings of the 5th international conference on predictor models in software engineering*, page 5. ACM.

[210] Tran, N. and Reed, D. A. (2001). Arima time series modeling and forecasting for adaptive i/o prefetching. In *Proceedings of the 15th International Conference on Supercomputing*, ICS '01, pages 473–485, New York, NY, USA. ACM.

[211] Tsay, R. S. (2005). *Analysis of financial time series*, volume 543. John Wiley & Sons.

[212] Turhan, B. and Bener, A. (2009). Analysis of naive bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, 68(2):278–290.

[213] Turhan, B., Bener, A., and Menzies, T. (2010). Regularities in learning defect predictors. In *International Conference on Product Focused Software Process Improvement*, pages 116–130. Springer.

[214] Ueda, Y., Kamiya, T., Kusumoto, S., and Inoue, K. (2002). On detection of gapped code clones using gap locations. In *Software Engineering Conference, 2002. Ninth Asia-Pacific*, pages 327–336. IEEE.

[215] Ullah, N., Morisio, M., and Vetro, A. (2015). Selecting the best reliability model to predict residual defects in open source software. *Computer*, 48(6):50–58.

[216] Volanschi, N. (2012). Safe clone-based refactoring through stereotype identification and iso-generation. In *Proceedings of the 6th International Workshop on Software Clones*, pages 50–56. IEEE Press.

[217] Von Krogh, G. (2003). Open-source software development. *MIT Sloan Management Review*, 44(3):14.

[218] Wahler, V., Seipel, D., von Gudenberg, J. W., and Fischer, G. (2004). Clone detection in source code by frequent itemset techniques. In *SCAM*, volume 4, pages 128–135.

[219] Wang, J. and Carroll, J. M. (2011). Behind linus's law: A preliminary analysis of open source software peer review practices in mozilla and python. In *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, pages 117–124. IEEE.

[220] Wang, Q., Yu, B., and Zhu, J. (2004). Extract rules from software quality prediction model based on neural network. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 191–195. IEEE.

[221] Wang, Q., Zhu, J., and Yu, B. (2007). Feature selection and clustering in software quality prediction. In *EASE*.

[222] Wei, W. W.-S. (1994). *Time series analysis*. Addison-Wesley publ Reading.

[223] Wiener, N. (1949). *Extrapolation, interpolation, and smoothing of stationary time series*, volume 7. MIT press Cambridge, MA.

[224] Wiper, M., Palacios, A., and Marín, J. (2012). Bayesian software reliability prediction using software metrics information. *Quality Technology & Quantitative Management*, 9(1):35–44.

[225] Wolf, T., Schroter, A., Damian, D., and Nguyen, T. (2009). Predicting build failures using social network analysis on developer communication. In *Proceedings of the 31st International Conference on Software Engineering*, pages 1–11. IEEE Computer Society.

[226] Wood, A. (1997). Software reliability growth models: assumptions vs. reality. In *Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on*, pages 136–141. IEEE.

[227] Wu, W., Zhang, W., Yang, Y., and Wang, Q. (2010). Time series analysis for bug number prediction. In *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on*, pages 589–596. IEEE.

[228] Xie, G., Chen, J., and Neamtiu, I. (2009). Towards a better understanding of software evolution: An empirical study on open source software. In *ICSM*, volume 9, pages 51–60.

[229] Xie, M. (1991). *Software reliability modelling*, volume 1. World Scientific.

[230] Xie, M. and Ho, S. (1999). Analysis of repairable system failure data using time series models. *Journal of Quality in Maintenance Engineering*, 5(1):50–61.

[231] Xie, M., Hong, G. Y., and Wohlin, C. (1999). Software reliability prediction incorporating information from a similar project. *Journal of Systems and Software*, 49(1):43–48.

[232] Xing, F., Guo, P., and Lyu, M. R. (2005). A novel method for early software quality prediction based on support vector machine. In *Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on*, pages 10–pp. IEEE.

[233] Xu, Z., Khoshgoftaar, T. M., and Allen, E. B. (2000). Prediction of software faults using fuzzy nonlinear regression modeling. In *High Assurance Systems Engineering, 2000, Fifth IEEE International Symposim on. HASE 2000*, pages 281–290. IEEE.

[234] Yang, B., Li, X., Xie, M., and Tan, F. (2010). A generic data-driven software reliability model with model mining technique. *Reliability Engineering & System Safety*, 95(6):671–678.

[235] Yang, B., Yao, L., and Huang, H.-Z. (2007). Early software quality prediction based on a fuzzy neural network model. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 1, pages 760–764. IEEE.

[236] Yang, B., Yin, Q., Xu, S., and Guo, P. (2008). Software quality prediction using affinity propagation algorithm. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1891–1896. IEEE.

[237] Yang, W. (1991). Identifying syntactic differences between two programs. *Software: Practice and Experience*, 21(7):739–755.

[238] Yeresime, S., Pati, J., and Rath, S. K. (2014). Review of software quality metrics for object-oriented methodology. In *Proceedings of International Conference on Internet Computing and Information Communications*, pages 267–278. Springer.

[239] Zaidi, S. J. H., Danial, S. N., and Usmani, B. A. (2008). Modeling inter-failure time series using neural networks. In *Multitopic Conference, 2008. INMIC 2008. IEEE International*, pages 409–411. IEEE.

[240] Zeitler, D. (1991). Realistic assumptions for software reliability models. In *Software Reliability Engineering, 1991. Proceedings., 1991 International Symposium on*, pages 67–74. IEEE.

[241] Zhang, G. P. (2003). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175.

[242] Zhang, H. (2008). An initial study of the growth of eclipse defects. In *Proceedings of the 2008 international working conference on Mining software repositories*, pages 141–144. ACM.

[243] Zhang, H. and Zhang, X. (2007). Comments on "data mining static code attributes to learn defect predictors". *IEEE Transactions on Software Engineering*, 33(9):635–637.

[244] Zhang, P. and Chang, Y.-t. (2012). Software fault prediction based on grey neural network. In *Natural Computation (ICNC), 2012 Eighth International Conference on*, pages 466–469. IEEE.

[245] Zhong, S., Khoshgoftaar, T. M., and Seliya, N. (2004). Unsupervised learning for expert-based software quality estimation. In *HASE*, pages 149–155.

[246] Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q. (2011a). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49.

[247] Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q. (2011b). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32 – 49.

[248] Zhou, Y. and Leung, H. (2006). Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Transactions on software engineering*, 32(10):771–789.

[249] Zimmermann, T., Nagappan, N., Gall, H., Giger, E., and Murphy, B. (2009). Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 91–100. ACM.

[250] Zissis, D., Xidias, E. K., and Lekkas, D. (2016). Real-time vessel behavior prediction. *Evolving Systems*, 7(1):29–40.