Appendix A

Publications

A.1 Journal Papers

(i) Pandey, S. K., & Triphathi, A. K. Predicting the next version of the Software System. DNNAttention: A Deep Neural Network and Attention based architecture for Cross Project Defect Number prediction. Knowledge-Based Systems, 197, 105924. **(SCI, IF: 8.03).**

(ii) Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2020). BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques. Expert Systems with Applications, 144, 113085. **(SCI, IF: 6.95).**

(iii) Pandey, S. K., & Tripathi, A. K. (2020). BCV-Predictor: A bug count vector predictor of a successive version of the software system. Knowledge-Based Systems, 197, 105924. **(SCI, IF: 8.03).**

(iv) Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2021). Machine Learning Based Methods for Software Fault Prediction: A Survey. Expert Systems with Applications, 114595. **(SCI, IF: 6.95).**

(v) Pandey, Sushant Kumar, and Anil Kumar Tripathi. "An empirical study toward dealing with noise and class imbalance issues in software defect prediction." Soft Computing (2021): 1-28. **(SCI, IF: 3.64).**

(vi) Pandey, Sushant Kumar, Deevashwer Rathee, and Anil Kumar Tripathi. "Software defect prediction using K-PCA and various kernel-based extreme learning machine: an empirical study." IET Software 14.7 (2020): 768-782. **(SCI, IF: 1.258).**

A.2 Conference Papers

(i) Pandey, Sushant Kumar, and Anil Kumar Tripathi. "Class Imbalance Issue in Software Defect Prediction Models by various Machine Learning Techniques: An Empirical Study." 2021 8th International Conference on Smart Computing and Communications (ICSCC). IEEE, 2021.

(ii) Pandey, S. K., Mishra, R. B., & Triphathi, A. K. (ICCIDS-2018). Software bug prediction prototype using bayesian network classifier: A comprehensive model. Procedia computer science, 132, 1412-1421.

A.3 Communicated Papers

(i) Pandey, S. K., Agarwal, Adit., & Triphathi, A. K. Predicting the next version of the Software System. ACM Transactions on Knowledge Discovery from Data **(Under Review)**.

Bibliography

[1] Abaei, G., Selamat, A., and Fujita, H. (2015). An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction. *Knowledge-Based Systems*, 74:28–39.

[2] Abdi, H. (2007). Bonferroni and šidák corrections for multiple comparisons. *Encyclopedia of measurement and statistics*, 3:103–107.

[3] Abdi, L. and Hashemi, S. (2015). To combat multi-class imbalanced problems by means of over-sampling techniques. *IEEE transactions on Knowledge and Data Engineering*, 28(1):238–251.

[4] Afzal, W., Torkar, R., and Feldt, R. (2008). Prediction of fault count data using genetic programming. In *2008 IEEE International Multitopic Conference*, pages 349–356. IEEE.

[5] Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. 6(1):37–66.

[6] Al Shalabi, L., Shaaban, Z., and Kasasbeh, B. (2006). Data mining: A preprocessing engine. *Journal of Computer Science*, 2(9):735–739.

[7] Aljamaan, H. I. and Elish, M. O. (2009). An empirical study of bagging and boosting ensembles for identifying faulty classes in object-oriented software. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 187–194. IEEE.

[8] Allen, E. (2002). *Bug patterns in Java.* APress.

[9] Altinger, H., Herbold, S., Schneemann, F., Grabowski, J., and Wotawa, F. (2017). Performance tuning for automotive software fault prediction. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 526–530. IEEE.

[10] Aman, H., Amasaki, S., Sasaki, T., and Kawahara, M. (2015). Lines of comments as a noteworthy metric for analyzing fault-proneness in methods. *IEICE TRANSACTIONS on Information and Systems*, 98(12):2218–2228.

[11] Amasaki, S. (2018). Cross-version defect prediction using cross-project defect prediction approaches: Does it work? In *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 32–41.

[12] Arar, Ö. F. and Ayan, K. (2015). Software defect prediction using cost-sensitive neural network. *Applied Soft Computing*, 33:263–277.

[13] Arar, Ö. F. and Ayan, K. (2017). A feature dependent naive bayes approach and its application to the software defect prediction problem. *Applied Soft Computing*, 59:197–209.

[14] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[15] Basili, V. R. and Zelkowitz, M. V. (1978). Analyzing medium-scale software development. In *Proceedings of the 3rd international conference on Software engineering*, pages 116–123. IEEE Press.

[16] Bell, R. M., Ostrand, T. J., and Weyuker, E. J. (2006). Looking for bugs in all the right places. In *Proceedings of the 2006 international symposium on Software testing and analysis*, pages 61–72. ACM.

[17] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

[18] Bennin, K. E., Keung, J., Monden, A., Phannachitta, P., and Mensah, S. (2017). The significant effects of data sampling approaches on software defect prioritization and classification. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 364–373. IEEE Press.

[19] Biçer, M. S. and Diri, B. (2015). Predicting defect prone modules in web applications. In *International Conference on Information and Software Technologies*, pages 577–591. Springer.

[20] Biçer, M. S. and Diri, B. (2016). Defect prediction for cascading style sheets. *Applied Soft Computing*, 49:1078–1084.

[21] Boehm, B. and Basili, V. R. (2007). Software defect reduction top 10 list. *Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research*, 34(1):75.

[22] Bowes, D., Hall, T., Harman, M., Jia, Y., Sarro, F., and Wu, F. (2016). Mutation-aware fault prediction. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pages 330–341. ACM.

[23] Bowes, D., Hall, T., and Petrić, J. (2018). Software defect prediction: do different classifiers find the same defects? *Software Quality Journal*, 26(2):525–552.

[24] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[25] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[26] Buyukyilmaz, M. and Cibikdiken, A. O. (2016). Voice gender recognition using deep learning. In *2016 International Conference on Modeling, Simulation and Optimization Technologies and Applications (MSOTA2016)*. Atlantis Press.

[27] Cao, X., Wipf, D., Wen, F., Duan, G., and Sun, J. (2013). A practical transfer learning algorithm for face verification. In *Proceedings of the IEEE international conference on computer vision*, pages 3208–3215.

[28] Catal, C. (2011). Software fault prediction: A literature review and current trends. *Expert systems with applications*, 38(4):4626–4636.

[29] Catal, C. and Diri, B. (2009a). Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences*, 179(8):1040–1058.

[30] Catal, C. and Diri, B. (2009b). A systematic review of software fault prediction studies. *Expert systems with applications*, 36(4):7346–7354.

[31] Charte, F., Rivera, A. J., del Jesus, M. J., and Herrera, F. (2015). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3–16.

[32] Chatterjee, S. and Maji, B. (2016). A new fuzzy rule based algorithm for estimating software faults in early phase of development. *Soft Computing*, 20(10):4023–4035.

[33] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[34] Chen, D., Chen, X., Li, H., Xie, J., and Mu, Y. (2019a). Deepcpdp: Deep learning based cross-project defect prediction. *IEEE Access*, 7:184832–184848.

[35] Chen, M. and Ma, Y. (2015). An empirical study on predicting defect numbers. In *SEKE*, pages 397–402.

[36] Chen, X. and Lawrence Zitnick, C. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431.

[37] Chen, X., Zhang, D., Zhao, Y., Cui, Z., and Ni, C. (2019b). Software defect number prediction: Unsupervised vs supervised methods. *Information and Software Technology*, 106:161–181.

[38] Chen, X.-W. and Lin, X. (2014). Big data deep learning: challenges and perspectives. *IEEE access*, 2:514–525.

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

[40] Christodoulidis, S., Anthimopoulos, M., Ebner, L., Christe, A., and Mougiakakou, S. (2016). Multisource transfer learning with convolutional neural networks for lung pattern analysis. *IEEE journal of biomedical and health informatics*, 21(1):76–84.

[41] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[42] CireşAn, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural networks*, 32:333–338.

[43] Cliff, N. (2014). *Ordinal methods for behavioral data analysis*. Psychology Press.

[44] Cochran, W. G. (2007). *Sampling techniques*. John Wiley & Sons.

[45] Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198. ACM.

[46] Czibula, G., Marian, Z., and Czibula, I. G. (2014). Software defect prediction using relational association rule mining. *Information Sciences*, 264:260–278.

[47] Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2007). Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200.

[48] Dam, H. K., Pham, T., Ng, S. W., Tran, T., Grundy, J., Ghose, A., Kim, T., and Kim, C.-J. (2018). A deep tree-based model for software defect prediction. *arXiv preprint arXiv:1802.00921*.

[49] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

[50] Deng, L., Hinton, G., and Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603. IEEE.

[51] Di Martino, S., Ferrucci, F., Gravino, C., and Sarro, F. (2011). A genetic algorithm to configure support vector machines for predicting fault-prone components. In *International Conference on Product Focused Software Process Improvement*, pages 247–261. Springer.

[52] Dietterich, T. G. et al. (2002). Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125.

[53] Dittman, D. J., Khoshgoftaar, T. M., and Napolitano, A. (2015). The effect of data sampling when using random forest on imbalanced bioinformatics data. In *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*, pages 457–463. IEEE.

[54] Do, C. B. and Ng, A. Y. (2006). Transfer learning for text classification. In *Advances in Neural Information Processing Systems*, pages 299–306.

[55] Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732.

[56] Duggal, H. and Singh, P. (2012). Comparative study of the performance of m5-rules algorithm with different algorithms.

[57] 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.

[58] Eaton, E. et al. (2011). Selective transfer between learning tasks using task-based boosting. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

[59] Fagundes, R. A., Souza, R. M., and Cysneiros, F. J. (2016). Zero-inflated prediction model in software-fault data. *IET Software*, 10(1):1–9.

[60] Fakoor, R., Ladhak, F., Nazi, A., and Huber, M. (2013). Using deep learning to enhance cancer diagnosis and classification. In *Proceedings of the international conference on machine learning*, volume 28. ACM New York, USA.

[61] Fan, G., Diao, X., Yu, H., Yang, K., and Chen, L. (2019). Software defect prediction via attention-based recurrent neural network. *Scientific Programming*, 2019.

[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] Fenton, N. E. and Neil, M. (2000). Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 357–370. Citeseer.

[64] Firat, O., Cho, K., and Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

[65] Frank, E. and Bouckaert, R. R. (2009). Conditional density estimation with class probability estimators. In *Asian Conference on Machine Learning*, pages 65–81. Springer.

[66] Freund, Y., Mansour, Y., and Schapire, R. E. (2001). Why averaging classifiers can protect against overfitting. In *AISTATS*. Citeseer.

[67] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

[68] Friedman, J., Hastie, T., Tibshirani, R., et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407.

[69] Fung, G. P. C., Yu, J. X., Lu, H., and Yu, P. S. (2005). Text classification without negative examples revisit. *IEEE transactions on Knowledge and Data Engineering*, 18(1):6–20.

[70] Ganai, M. K. and Wang, C. (2010). Interval analysis for concurrent trace programs using transaction sequence graphs. In *International Conference on Runtime Verification*, pages 253–269. Springer.

[71] 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.

[72] Gao, L., Guo, Z., Zhang, H., Xu, X., and Shen, H. T. (2017). Video captioning with attention-based lstm and semantic consistency. *IEEE Transactions on Multimedia*, 19(9):2045–2055.

[73] García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064.

[74] Garner, S. R. et al. (1995). Weka: The waikato environment for knowledge analysis. In *Proceedings of the New Zealand computer science research students conference*, pages 57–64.

[75] Gelfand, S. B., Ravishankar, C., and Delp, E. J. (1989). An iterative growing and pruning algorithm for classification tree design. In *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 818–823. IEEE.

[76] Ghotra, B., McIntosh, S., and Hassan, A. E. (2015). Revisiting the impact of classification techniques on the performance of defect prediction models. In

2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 1, pages 789–800. IEEE.

[77] Ghotra, B., McIntosh, S., and Hassan, A. E. (2017). A large-scale study of the impact of feature selection techniques on defect classification models. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 146–157. IEEE.

[78] Gong, L., Jiang, S., Bo, L., Jiang, L., and Qian, J. (2019). A novel class-imbalance learning approach for both within-project and cross-project defect prediction. *IEEE Transactions on Reliability*.

[79] Graves, T. L., Karr, A. F., Marron, J. S., and Siy, H. (2000). Predicting fault incidence using software change history. *IEEE Transactions on software engineering*, 26(7):653–661.

[80] Gray, D., Bowes, D., Davey, N., Sun, Y., and Christianson, B. (2011). The misuse of the nasa metrics data program data sets for automated software defect prediction. In *Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*, pages 96–103. IET.

[81] Gray, D., Bowes, D., Davey, N., Sun, Y., and Christianson, B. (2012). Reflections on the nasa mdp data sets. *IET software*, 6(6):549–558.

[82] Greene, W. H. (2003). *Econometric analysis*. Pearson Education India.

[83] Gulsun, M. A., Zheng, Y., Sharma, P., Georgescu, B., and Comaniciu, D. (2017). Method and system for vascular disease detection using recurrent neural networks. US Patent 9,767,557.

[84] Guo, L., Ma, Y., Cukic, B., and Singh, H. (2004). Robust prediction of fault-proneness by random forests. In *15th international symposium on software reliability engineering*, pages 417–428. IEEE.

[85] Hall, T., Beecham, S., Bowes, D., Gray, D., and Counsell, S. (2011). A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, 38(6):1276–1304.

[86] Hall, T., Zhang, M., Bowes, D., and Sun, Y. (2014). Some code smells have a significant but small effect on faults. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 23(4):33.

[87] Halstead, M. H. (1977). Elements of software science.

[88] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[89] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

[90] Heffernan, R., Paliwal, K., Lyons, J., Dehzangi, A., Sharma, A., Wang, J., Sattar, A., Yang, Y., and Zhou, Y. (2015). Improving prediction of secondary structure, local backbone angles and solvent accessible surface area of proteins by iterative deep learning. *Scientific reports*, 5(1):1–11.

[91] Herbold, S., Trautsch, A., and Grabowski, J. (2017). A comparative study to benchmark cross-project defect prediction approaches. *IEEE Transactions on Software Engineering*, 44(9):811–833.

[92] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[93] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

[94] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[95] Hong, S., Noh, H., and Han, B. (2015). Decoupled deep neural network for semi-supervised semantic segmentation. In *Advances in neural information processing systems*, pages 1495–1503.

[96] Iba, W. and Langley, P. (1992). Induction of one-level decision trees. In *Machine Learning Proceedings 1992*, pages 233–240. Elsevier.

[97] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[98] Irie, K., Tüske, Z., Alkhouli, T., Schlüter, R., and Ney, H. (2016). Lstm, gru, highway and a bit of attention: An empirical overview for language modeling in speech recognition. In *Interspeech*, pages 3519–3523.

[99] Janes, A., Scotto, M., Pedrycz, W., Russo, B., Stefanovic, M., and Succi, G. (2006). Identification of defect-prone classes in telecommunication software systems using design metrics. *Information sciences*, 176(24):3711–3734.

[100] Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.

[101] Ji, H., Huang, S., Wu, Y., Hui, Z., and Zheng, C. (2019). A new weighted naive bayes method based on information diffusion for software defect prediction. *Software Quality Journal*, pages 1–46.

[102] Jia, W., Muhammad, K., Wang, S.-H., and Zhang, Y.-D. (2019). Five-category classification of pathological brain images based on deep stacked sparse autoencoder. *Multimedia Tools and Applications*, 78(4):4045–4064.

[103] Jin, C. and Jin, S.-W. (2015). Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization. *Applied Soft Computing*, 35:717–725.

[104] Jing, X.-Y., Wu, F., Dong, X., and Xu, B. (2016). An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems. *IEEE Transactions on Software Engineering*, 43(4):321–339.

[105] Jing, X.-Y., Ying, S., Zhang, Z.-W., Wu, S.-S., and Liu, J. (2014). Dictionary learning based software defect prediction. In *Proceedings of the 36th International Conference on Software Engineering*, pages 414–423. ACM.

[106] Joachims, T. (1996). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science.

[107] Jorgensen, M. (1995). Experience with the accuracy of software maintenance task effort prediction models. *IEEE Transactions on software engineering*, 21(8):674–681.

[108] Jureczko, M. and Madeyski, L. (2010). Towards identifying software project clusters with regard to defect prediction. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, pages 1–10.

[109] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

[110] Kamei, Y., Fukushima, T., McIntosh, S., Yamashita, K., Ubayashi, N., and Hassan, A. E. (2016). Studying just-in-time defect prediction using cross-project models. *Empirical Software Engineering*, 21(5):2072–2106.

[111] Kamei, Y., Monden, A., Matsumoto, S., Kakimoto, T., and Matsumoto, K.-i. (2007). The effects of over and under sampling on fault-prone module detection. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pages 196–204. IEEE.

[112] Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A., and Ubayashi, N. (2013). A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering*, 39(6):757–773.

[113] Kanmani, S., Uthariaraj, V. R., Sankaranarayanan, V., and Thambidurai, P. (2007). Object-oriented software fault prediction using neural networks. *Information and software technology*, 49(5):483–492.

[114] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

[115] Kaur, A., Kaur, K., and Kaur, H. (2016). Application of machine learning on process metrics for defect prediction in mobile application. In *Information Systems Design and Intelligent Applications*, pages 81–98. Springer.

[116] Khoshgoftaar, T. M. and Allen, E. B. (1999). Logistic regression modeling of software quality. *International Journal of Reliability, Quality and Safety Engineering*, 6(04):303–317.

[117] Khoshgoftaar, T. M. and Allen, E. B. (2001). Controlling overfitting in classification-tree models of software quality. *Empirical Software Engineering*, 6(1):59–79.

[118] Khoshgoftaar, T. M. and Gao, K. (2007). Count models for software quality estimation. *IEEE Transactions on Reliability*, 56(2):212–222.

[119] Khoshgoftaar, T. M., Gao, K., and Seliya, N. (2010). Attribute selection and imbalanced data: Problems in software defect prediction. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 137–144. IEEE.

[120] Khoshgoftaar, T. M., Geleyn, E., and Nguyen, L. (2003). Empirical case studies of combining software quality classification models. In *Third International Conference on Quality Software, 2003. Proceedings.*, pages 40–49. IEEE.

[121] Khoshgoftaar, T. M., Pandya, A. S., and More, H. B. (1992). A neural network approach for predicting software development faults. In *Proceedings Third International Symposium on Software Reliability Engineering*, pages 83–89. IEEE.

[122] Khuat, T. T. and Le, M. H. (2019). Ensemble learning for software fault prediction problem with imbalanced data. *International Journal of Electrical & Computer Engineering (2088-8708)*, 9.

[123] Kim, S., Whitehead Jr, E. J., and Zhang, Y. (2008). Classifying software changes: Clean or buggy? *IEEE Transactions on Software Engineering*, 34(2):181–196.

[124] Kim, S., Zhang, H., Wu, R., and Gong, L. (2011). Dealing with noise in defect prediction. In *2011 33rd International Conference on Software Engineering (ICSE)*, pages 481–490. IEEE.

[125] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[126] Kittler, J., Hater, M., and Duin, R. P. (1996). Combining classifiers. In *Proceedings of 13th international conference on pattern recognition*, volume 2, pages 897–901. IEEE.

[127] Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

[128] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[129] Kumar, L., Sripada, S. K., Sureka, A., and Rath, S. K. (2017). Effective fault prediction model developed using least square support vector machine (lssvm). *Journal of Systems and Software*.

[130] Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207.

[131] Lam, F. and Longnecker, M. (1983). A modified wilcoxon rank sum test for paired data. *Biometrika*, 70(2):510–513.

[132] Lambert, D. (1992). Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14.

[133] Laradji, I. H., Alshayeb, M., and Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, 58:388–402.

[134] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[135] Lessmann, S., Baesens, B., Mues, C., and Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4):485–496.

[136] Li, B., Shen, B., Wang, J., Chen, Y., Zhang, T., and Wang, J. (2014). A scenario-based approach to predicting software defects using compressed c4. 5 model. In *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*, pages 406–415. IEEE.

[137] 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.

[138] Li, W., Huang, Z., and Li, Q. (2016). Three-way decisions based software defect prediction. *Knowledge-Based Systems*, 91:263–274.

[139] Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

[140] Lim, H. and Goel, A. L. (2008). Software effort prediction. *Wiley Encyclopedia of Computer Science and Engineering*.

[141] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

[142] Liu, C., Yang, D., Xia, X., Yan, M., and Zhang, X. (2019). A two-phase transfer learning model for cross-project defect prediction. *Information and Software Technology*, 107:125–136.

[143] Liu, J., Pan, Y., Li, M., Chen, Z., Tang, L., Lu, C., and Wang, J. (2018). Applications of deep learning to mri images: A survey. *Big Data Mining and Analytics*, 1(1):1–18.

[144] Liu, J., Wang, G., Duan, L.-Y., Abdiyeva, K., and Kot, A. C. (2017a). Skeleton-based human action recognition with global context-aware attention lstm networks. *IEEE Transactions on Image Processing*, 27(4):1586–1599.

[145] Liu, J., Wang, G., Hu, P., Duan, L.-Y., and Kot, A. C. (2017b). Global context-aware attention lstm networks for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1647–1656.

[146] Liu, Y., Khoshgoftaar, T. M., and Seliya, N. (2010). Evolutionary optimization of software quality modeling with multiple repositories. *IEEE Transactions on Software Engineering*, 36(6):852–864.

[147] Lu, S., Park, S., Hu, C., Ma, X., Jiang, W., Li, Z., Popa, R. A., and Zhou, Y. (2007). Muvi: automatically inferring multi-variable access correlations and detecting related semantic and concurrency bugs. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 103–116. ACM.

[148] Ma, B., Zhang, H., Chen, G., Zhao, Y., and Baesens, B. (2014a). Investigating associative classification for software fault prediction: an experimental perspective. *International Journal of Software Engineering and Knowledge Engineering*, 24(01):61–90.

[149] Ma, W., Chen, L., Yang, Y., Zhou, Y., and Xu, B. (2016). Empirical analysis of network measures for effort-aware fault-proneness prediction. *Information and Software Technology*, 69:50–70.

[150] Ma, Y., Luo, G., Zeng, X., and Chen, A. (2012). Transfer learning for cross-company software defect prediction. *Information and Software Technology*, 54(3):248–256.

[151] Ma, Y., Pan, W., Zhu, S., Yin, H., and Luo, J. (2014b). An improved semi-supervised learning method for software defect prediction. *Journal of Intelligent & Fuzzy Systems*, 27(5):2473–2480.

[152] Ma, Y., Qin, K., and Zhu, S. (2014c). Discrimination analysis for predicting defect-prone software modules. *Journal of Applied Mathematics*, 2014.

[153] 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.

[154] Mahmood, Z., Bowes, D., Lane, P. C., and Hall, T. (2015). What is the impact of imbalance on software defect prediction performance? In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*, page 4. ACM.

[155] Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, 27:504–518.

[156] Manjula, C. and Florence, L. (2018). Deep neural network based hybrid approach for software defect prediction using software metrics. *Cluster Computing*, pages 1–17.

[157] Martini, A. T., Farrukh, M., and Ge, H. (2018). Recognition of ironic sentences in twitter using attention-based lstm. *International Journal of Advanced Computer Science and Applications*, 9(8).

[158] Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

[159] Mauša, G. and Grbac, T. G. (2017). Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study. *Applied soft computing*, 55:331–351.

[160] McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on software Engineering*, (4):308–320.

[161] Menotti, D., Chiachia, G., Pinto, A., Schwartz, W. R., Pedrini, H., Falcao, A. X., and Rocha, A. (2015). Deep representations for iris, face, and fingerprint spoofing detection. *IEEE Transactions on Information Forensics and Security*, 10(4):864–879.

[162] Menzies, T., Greenwald, J., and Frank, A. (2006). Data mining static code attributes to learn defect predictors. *IEEE transactions on software engineering*, 33(1):2–13.

[163] Miao, Y., Gowayyed, M., and Metze, F. (2015). Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE.

[164] Mısırlı, A. T., Bener, A. B., and Turhan, B. (2011). An industrial case study of classifier ensembles for locating software defects. *Software Quality Journal*, 19(3):515–536.

[165] Mohamed, W. N. H. W., Salleh, M. N. M., and Omar, A. H. (2012). A comparative study of reduced error pruning method in decision tree algorithms. In *Control System, Computing and Engineering (ICCSCE), 2012 IEEE International Conference on*, pages 392–397. IEEE.

[166] Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419.

[167] Moser, R., Pedrycz, W., and Succi, G. (2008). A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Proceedings of the 30th international conference on Software engineering*, pages 181–190. ACM.

[168] Murphy, K. P. et al. (2006). Naive bayes classifiers. *University of British Columbia*, 18.

[169] Musa, J. D., Iannino, A., and Okumoto, K. (1990). Software reliability. *Advances in computers*, 30:85–170.

[170] Nagappan, N. and Ball, T. (2005). Static analysis tools as early indicators of pre-release defect density. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 580–586. IEEE.

[171] Nagappan, N., Murphy, B., and Basili, V. (2008). The influence of organizational structure on software quality. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 521–530. IEEE.

[172] Nagappan, N. and Wang, X. (2016). Hydra: Massively compositional model for cross-project defect prediction. *IEEE Transactions on software Engineering*, 42(10):977.

[173] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1.

[174] Nam, J. and Kim, S. (2015). Clami: Defect prediction on unlabeled datasets (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 452–463. IEEE.

[175] Nam, J., Pan, S. J., and Kim, S. (2013). Transfer defect learning. In *2013 35th international conference on software engineering (ICSE)*, pages 382–391. IEEE.

[176] Ng, H.-W., Nguyen, V. D., Vonikakis, V., and Winkler, S. (2015). Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the 2015 ACM on international conference on multimodal interaction*, pages 443–449. ACM.

[177] Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H. G., and Ogata, T. (2015). Audio-visual speech recognition using deep learning. *Applied Intelligence*, 42(4):722–737.

[178] Nunez, J. C., Cabido, R., Pantrigo, J. J., Montemayor, A. S., and Velez, J. F. (2018). Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognition*, 76:80–94.

[179] Okutan, A. and Yıldız, O. T. (2014). Software defect prediction using bayesian networks. *Empirical Software Engineering*, 19(1):154–181.

[180] Oshiro, T. M., Perez, P. S., and Baranauskas, J. A. (2012). How many trees in a random forest? In *International workshop on machine learning and data mining in pattern recognition*, pages 154–168. Springer.

[181] Ostrand, T. J., Weyuker, E. J., and Bell, R. M. (2004). Where the bugs are. In *ACM SIGSOFT Software Engineering Notes*, volume 29, pages 86–96. ACM.

[182] 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.

[183] Oyedotun, O. K. and Khashman, A. (2017). Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications*, 28(12):3941–3951.

[184] Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2010). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

[185] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[186] Panichella, A., Oliveto, R., and De Lucia, A. (2014). 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)*, pages 164–173. IEEE.

[187] Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2.

[188] Pelayo, L. and Dick, S. (2007). Applying novel resampling strategies to software defect prediction. In *NAFIPS 2007-2007 Annual Meeting of the North American Fuzzy Information Processing Society*, pages 69–72. IEEE.

[189] Pelayo, L. and Dick, S. (2012). Evaluating stratification alternatives to improve software defect prediction. *IEEE transactions on reliability*, 61(2):516–525.

[190] Peng, Y., Kou, G., Wang, G., Wu, W., and Shi, Y. (2011). Ensemble of software defect predictors: an ahp-based evaluation method. *International Journal of Information Technology & Decision Making*, 10(01):187–206.

[191] Pérez, J. M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I., and Martín, J. I. (2007). Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, 28(4):414–422.

[192] Peters, F., Menzies, T., and Marcus, A. (2013). Better cross company defect prediction. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 409–418. IEEE.

[193] Peto, R. and Peto, J. (1972). Asymptotically efficient rank invariant test procedures. *Journal of the Royal Statistical Society: Series A (General)*, 135(2):185–198.

[194] Petrić, J., Bowes, D., Hall, T., Christianson, B., and Baddoo, N. (2016). The jinx on the nasa software defect data sets. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, page 13. ACM.

[195] Polikar, R. (2012). Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer.

[196] Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

[197] Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pages 2352–2360.

[198] Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE transactions on Software Engineering*, (4):345–361.

[199] Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13.

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

[201] Rathore, S. S. and Kumar, S. (2016). A decision tree regression based approach for the number of software faults prediction. *ACM SIGSOFT Software Engineering Notes*, 41(1):1–6.

[202] Rathore, S. S. and Kumar, S. (2017a). Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems. *Knowledge-Based Systems*, 119:232–256.

[203] Rathore, S. S. and Kumar, S. (2017b). Towards an ensemble based system for predicting the number of software faults. *Expert Systems with Applications*, 82:357–382.

[204] Rathore, S. S. and Kumar, S. (2018). An approach for the prediction of number of software faults based on the dynamic selection of learning techniques. *IEEE Transactions on Reliability*, 68(1):216–236.

[205] Rathore, S. S. and Kumar, S. (2019a). Nonlinear rule based ensemble methods for the prediction of number of faults. In *Fault Prediction Modeling for the Prediction of Number of Software Faults*, pages 59–69. Springer.

[206] Rathore, S. S. and Kumar, S. (2019b). A study on software fault prediction techniques. *Artificial Intelligence Review*, 51(2):255–327.

[207] Rätsch, G., Onoda, T., and Müller, K. R. (1998). An improvement of adaboost to avoid overfitting. In *Proc. of the Int. Conf. on Neural Information Processing*. Citeseer.

[208] Riaz, M., Mendes, E., and Tempero, E. (2009). A systematic review of software maintainability prediction and metrics. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 367–377. IEEE Computer Society.

[209] Ricky, M. Y., Purnomo, F., and Yulianto, B. (2016). Mobile application software defect prediction. In *Service-Oriented System Engineering (SOSE), 2016 IEEE Symposium on*, pages 307–313. IEEE.

[210] Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

[211] Rolnick, D., Veit, A., Belongie, S., and Shavit, N. (2017). Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

[Ruck et al.] Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E., and Suter, B. W. The multilayer perceptron as an approximation to a bayes optimal discriminant function.

[213] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[214] Sasidharan, R. and Sriram, P. (2014). Hyper-quadtree-based k-means algorithm for software fault prediction. In *Computational Intelligence, Cyber Security and Computational Models*, pages 107–118. Springer.

[215] Sayyad Shirabad, J. and Menzies, T. (2005a). The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada.

[216] Sayyad Shirabad, J. and Menzies, T. (2005b). The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada.

[217] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[218] Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W., Bridgland, A., et al. (2019). Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (casp13). *Proteins: Structure, Function, and Bioinformatics*, 87(12):1141–1148.

[219] Severyn, A. and Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962.

[220] Shepperd, M., Song, Q., Sun, Z., and Mair, C. (2013). Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, 39(9):1208–1215.

[221] Shirabad, J. S. and Menzies, T. J. (2005). The promise repository of software engineering databases. *School of Information Technology and Engineering, University of Ottawa, Canada*, 24.

[222] Shivaji, S., Whitehead, E. J., Akella, R., and Kim, S. (2012). Reducing features to improve code change-based bug prediction. *IEEE Transactions on Software Engineering*, 39(4):552–569.

[223] Shivaji, S., Whitehead Jr, E. J., Akella, R., and Kim, S. (2009). Reducing features to improve bug prediction. In *2009 IEEE/ACM International Conference on Automated Software Engineering*, pages 600–604. IEEE.

[224] Shukla, S., Radhakrishnan, T., Muthukumaran, K., and Neti, L. B. M. (2018). Multi-objective cross-version defect prediction. *Soft Computing*, 22(6):1959–1980.

[225] Siers, M. J. and Islam, M. Z. (2015). Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. *Information Systems*, 51:62–71.

[226] Singh, P., Pal, N. R., Verma, S., and Vyas, O. P. (2016). Fuzzy rule-based approach for software fault prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(5):826–837.

[227] Singh, S. P., Kumar, A., Darbari, H., Singh, L., Rastogi, A., and Jain, S. (2017). Machine translation using deep learning: An overview. In *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pages 162–167. IEEE.

[228] Singh, V. and Chaturvedi, K. (2013). Improving the quality of software by quantifying the code change metric and predicting the bugs. In *International Conference on Computational Science and Its Applications*, pages 408–426. Springer.

[229] Singh, Y., Kaur, A., and Malhotra, R. (2009). Software fault proneness prediction using support vector machines. In *Proceedings of the world congress on engineering*, volume 1, pages 1–3.

[230] Smola, A. J. and Bartlett, P. L. (2001). Sparse greedy gaussian process regression. In *Advances in neural information processing systems*, pages 619–625.

[231] 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.

[232] Solis, F. J. and Wets, R. J.-B. (1981). Minimization by random search techniques. *Mathematics of operations research*, 6(1):19–30.

[233] Sommerville, I. (2011). Software engineering 9th edition. *ISBN-10*, 137035152:18.

[234] Song, J., Guo, Z., Gao, L., Liu, W., Zhang, D., and Shen, H. T. (2017). Hierarchical lstm with adjusted temporal attention for video captioning. *arXiv preprint arXiv:1706.01231*.

[235] Song, Q., Jia, Z., Shepperd, M., Ying, S., and Liu, J. (2010). A general software defect-proneness prediction framework. *IEEE transactions on software engineering*, 37(3):356–370.

[236] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[237] Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

[238] Strezoski, G., Stojanovski, D., Dimitrovski, I., and Madjarov, G. (2016). Hand gesture recognition using deep convolutional neural networks. In *International conference on ICT innovations*, pages 49–58. Springer.

[239] Su, B., yong Chen, H., Chen, P., Bian, G.-B., Liu, W., et al. (2020). Deep learning-based solar-cell manufacturing defect detection with complementary attention network. *IEEE Transactions on Industrial Informatics*.

[240] Su, C., Ju, S., Liu, Y., and Yu, Z. (2015). Improving random forest and rotation forest for highly imbalanced datasets. *Intelligent Data Analysis*, 19(6):1409–1432.

[241] Sun, Z., Song, Q., and Zhu, X. (2012). Using coding-based ensemble learning to improve software defect prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1806–1817.

[242] Sundermeyer, M., Schlüter, R., and Ney, H. (2012). Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

[243] Sushant, A. (2020). Meta-datasest and source code. `https://github.com/sushantkumar007007/sushantkumar007007-gmail.com`.

[244] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

[245] Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to sequence learning with neural networks. *Advances in NIPS*.

[246] Tang, S., Wu, Z., and Chen, K. (2017). Movie recommendation via blstm. In *International Conference on Multimedia Modeling*, pages 269–279. Springer.

[247] Tantithamthavorn, C. and Hassan, A. E. (2018). An experience report on defect modelling in practice: Pitfalls and challenges. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pages 286–295. ACM.

[248] Tong, C., Li, J., Lang, C., Kong, F., Niu, J., and Rodrigues, J. J. (2018a). An efficient deep model for day-ahead electricity load forecasting with stacked denoising auto-encoders. *Journal of Parallel and Distributed Computing*, 117:267–273.

[249] Tong, H., Liu, B., and Wang, S. (2018b). Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. *Information and Software Technology*, 96:94–111.

[250] Turabieh, H., Mafarja, M., and Li, X. (2019). Iterated feature selection algorithms with layered recurrent neural network for software fault prediction. *Expert Systems with Applications*, 122:27–42.

[251] Vashisht, V., Lal, M., and Sureshchandar, G. (2016). Defect prediction framework using neural networks for software enhancement projects. *British Journal of Mathematics & Computer Science*, 16(5):384–394.

[252] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[253] Vezhnevets, A. and Barinova, O. (2007). Avoiding boosting overfitting by removing confusing samples. In *European Conference on Machine Learning*, pages 430–441. Springer.

[254] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

[255] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

[256] Wagner, S. (2006). A literature survey of the quality economics of defect-detection techniques. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 194–203.

[257] Walczak, B. and Massart, D. (2001). Dealing with missing data: Part i. *Chemometrics and Intelligent Laboratory Systems*, 58(1):15–27.

[258] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. (2017). Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

[259] Wang, G., Li, W., Zuluaga, M. A., Pratt, R., Patel, P. A., Aertsen, M., Doel, T., David, A. L., Deprest, J., Ourselin, S., et al. (2018a). Interactive medical image segmentation using deep learning with image-specific fine tuning. *IEEE transactions on medical imaging*, 37(7):1562–1573.

[260] Wang, J. and Zhang, C. (2018). Software reliability prediction using a deep learning model based on the rnn encoder–decoder. *Reliability Engineering & System Safety*, 170:73–82.

[261] Wang, J. and Zhang, H. (2012). Predicting defect numbers based on defect state transition models. In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 191–200. IEEE.

[262] Wang, S., Liu, T., and Tan, L. (2016a). Automatically learning semantic features for defect prediction. In *Proceedings of the 38th International Conference on Software Engineering*, pages 297–308. ACM.

[263] Wang, S., Minku, L. L., and Yao, X. (2013). Online class imbalance learning and its applications in fault detection. *International Journal of Computational Intelligence and Applications*, 12(04):1340001.

[264] Wang, S., Peng, J., Ma, J., and Xu, J. (2016b). Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports*, 6(1):1–11.

[265] Wang, S. and Yao, X. (2012). Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130.

[266] Wang, S. and Yao, X. (2013). Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2):434–443.

[267] Wang, T., Li, W., Shi, H., and Liu, Z. (2011). Software defect prediction based on classifiers ensemble. *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, 8(16):4241–4254.

[268] Wang, Y., Huang, M., Zhu, X., and Zhao, L. (2016c). Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

[269] Wang, Z., Wang, J., and Wang, Y. (2018b). An intelligent diagnosis scheme based on generative adversarial learning deep neural networks and its application to planetary gearbox fault pattern recognition. *Neurocomputing*, 310:213–222.

[270] Wei, H., Hu, C., Chen, S., Xue, Y., and Zhang, Q. (2019). Establishing a software defect prediction model via effective dimension reduction. *Information Sciences*, 477:399–409.

[271] Weigend, A. (1994). On overfitting and the effective number of hidden units. In *Proceedings of the 1993 connectionist models summer school*, volume 1, pages 335–342.

[272] Weyuker, E. J., Ostrand, T. J., and Bell, R. M. (2010). Comparing the effectiveness of several modeling methods for fault prediction. *Empirical Software Engineering*, 15(3):277–295.

[273] Wilcoxon, F., Katti, S., and Wilcox, R. A. (1970). Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected tables in mathematical statistics*, 1:171–259.

[274] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[275] Woolson, R. (2007). Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials*, pages 1–3.

[276] Wu, C., Fan, W., He, Y., Sun, J., and Naoi, S. (2014). Handwritten character recognition by alternately trained relaxation convolutional neural network. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 291–296. IEEE.

[277] Xia, X., Lo, D., Pan, S. J., Nagappan, N., and Wang, X. (2016). Hydra: Massively compositional model for cross-project defect prediction. *IEEE Transactions on software Engineering*, 42(10):977–998.

[278] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

[279] Xu, Z., Li, S., Luo, X., Liu, J., Zhang, T., Tang, Y., Xu, J., Yuan, P., and Keung, J. (2019a). Tstss: A two-stage training subset selection framework for cross version defect prediction. *Journal of Systems and Software*, 154:59–78.

[280] Xu, Z., Liu, J., Luo, X., Yang, Z., Zhang, Y., Yuan, P., Tang, Y., and Zhang, T. (2019b). Software defect prediction based on kernel pca and weighted extreme learning machine. *Information and Software Technology*, 106:182–200.

[281] Xu, Z., Liu, J., Luo, X., and Zhang, T. (2018). Cross-version defect prediction via hybrid active learning with kernel principal component analysis. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 209–220. IEEE.

[282] Xu, Z., Liu, J., Yang, Z., An, G., and Jia, X. (2016). The impact of feature selection on defect prediction performance: An empirical comparison. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 309–320. IEEE.

[283] Yadav, H. B. and Yadav, D. K. (2015). A fuzzy logic based approach for phase-wise software defects prediction using software metrics. *Information and Software Technology*, 63:44–57.

[284] Yamada, S., Ohtera, H., and Narihisa, H. (1986). Software reliability growth models with testing-effort. *IEEE Transactions on Reliability*, 35(1):19–23.

[285] Yang, X., Lo, D., Xia, X., Zhang, Y., and Sun, J. (2015). Deep learning for just-in-time defect prediction. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*, pages 17–26. IEEE.

[286] Yang, X. and Wen, W. (2018). Ridge and lasso regression models for cross-version defect prediction. *IEEE Transactions on Reliability*, 67(3):885–896.

[287] Ye, X., Bunescu, R., and Liu, C. (2015). Mapping bug reports to relevant files: A ranking model, a fine-grained benchmark, and feature evaluation. *IEEE Transactions on Software Engineering*, 42(4):379–402.

[288] Yu, J. (2019). A selective deep stacked denoising autoencoders ensemble with negative correlation learning for gearbox fault diagnosis. *Computers in Industry*, 108:62–72.

[289] Yu, L. (2012). Using negative binomial regression analysis to predict software faults: A study of apache ant.

[290] Yuan, Y., Chao, M., and Lo, Y.-C. (2017). Automatic skin lesion segmentation using deep fully convolutional networks with jaccard distance. *IEEE transactions on medical imaging*, 36(9):1876–1886.

[291] Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[292] Zhang, Z.-W., Jing, X.-Y., and Wang, T.-J. (2017). Label propagation based semi-supervised learning for software defect prediction. *Automated Software Engineering*, 24(1):47–69.

[293] Zhao, Y., Yang, Y., Lu, H., Liu, J., Leung, H., Wu, Y., Zhou, Y., and Xu, B. (2017). Understanding the value of considering client usage context in package cohesion for fault-proneness prediction. *Automated Software Engineering*, 24(2):393–453.

[294] Zhao, Y., Yang, Y., Lu, H., Zhou, Y., Song, Q., and Xu, B. (2015). An empirical analysis of package-modularization metrics: Implications for software fault-proneness. *Information and Software Technology*, 57:186–203.

[295] Zheng, J. (2010). Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6):4537–4543.

[296] Zhou, C., Sun, C., Liu, Z., and Lau, F. (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

[297] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.

[298] Zhou, Y. and Leung, H. (2007). Predicting object-oriented software maintainability using multivariate adaptive regression splines. *Journal of Systems and Software*, 80(8):1349–1361.

[299] Zhu, G., Zhang, L., Mei, L., Shao, J., Song, J., and Shen, P. (2016). Large-scale isolated gesture recognition using pyramidal 3d convolutional networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 19–24. IEEE.

[300] Zhu, Y., Chen, Y., Lu, Z., Pan, S. J., Xue, G.-R., Yu, Y., and Yang, Q. (2011). Heterogeneous transfer learning for image classification. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

[301] 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 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 91–100.